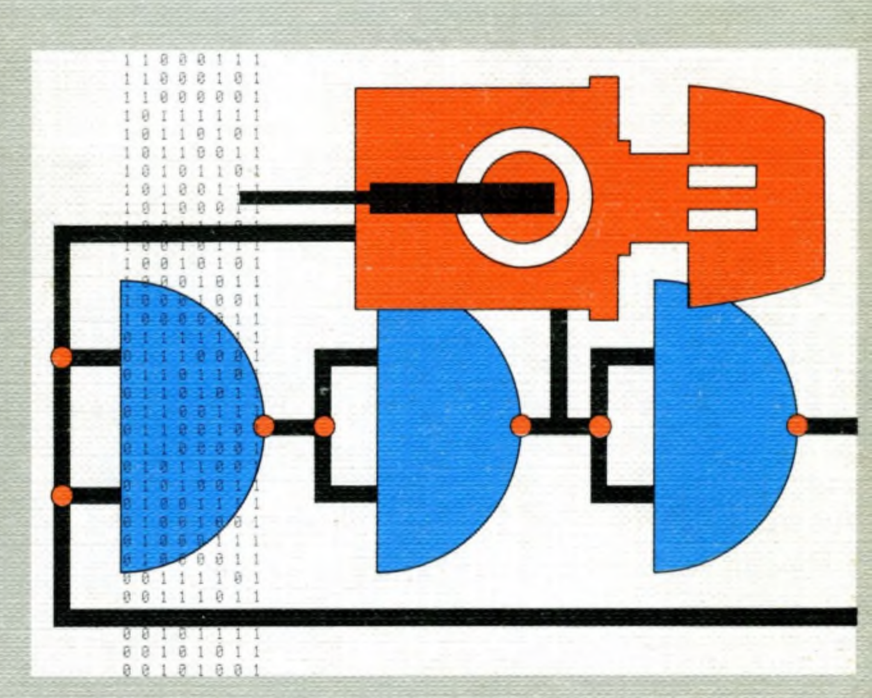
Basteln für Ingenieure

Klein
Mikrocomputersysteme

Selbstbau, Programmierung, Anwendung

Aufbau-Prinzip
Grundausstattung
Prüfstift für TTL-Schaltungen
Universalzähler
Mikroprogrammsteuereinheit
Ausgabesysteme
Schaltbild eines
Datensichtgerätes
Hilfsschaltung für Tastatur
Drucker
Steuerteil für BUS-Interface
Eingabesysteme
Zehner-Tastatur
Perifere Speicher
Wirkungsweise und Aufbau des
Mikrocomputers
Speicher mit RAM und PROM
Anwendung und Verknüpfung
der aufgebauten Einheiten
Aufstellen von Funktionstabellen
Einsatz von Programmiersprachen
auf dem Mikrocomputer



Franzis



Basteln für Ingenieure

Rolf-Dieter Klein

Mikrocomputersysteme

Selbstbau, Programmierung, Anwendung

Mit 133 Abbildungen und 11 Tabellen



CIP-Kurztitelaufnahme der Deutschen Bibliothek

Klein, Rolf-Dieter
Mikrocomputersysteme: Selbstbau, Programmierung, Anwendung. – 1. Aufl. – München: Franzis, 1978.
(Basteln für Ingenieure)
ISBN 3-7723-6381-4

1978

Franzis-Verlag GmbH, München

Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2
Printed in Germany · Imprimé en Allemagne

ISBN 3-7723-6381-4

Vorwort

Das vorliegende Buch soll eine ganze Reihe von Aufgaben erfüllen helfen. Die neue, sich zur Zeit durchsetzende Technik der Mikroprozessoren erfordert ein völliges Umstellen bei der Bearbeitung von Aufgaben, sowohl hinsichtlich der Gestaltung der Geräte (hardware), als auch bei deren Einsatz. Die Schaltungstechnik, die bisher trotz integrierter Kreise weitgehend erfunden, d. h. entwickelt werden mußte, ist in den modernen, höchst integrierten (LSI) Kreisen bereits enthalten. Es bleibt die Aufgabe, sich Zugang zu den Schaltungen zu verschaffen, d. h. Ein- und Ausgangseinheiten richtig abzuschließen und, je nach Zweck des Gerätes, den vorhandenen Befehlsvorrat einzusetzen (software). Es bleibt also, um im Jargon der Computer-Fachleute zu sprechen, vorwiegend Software zu produzieren.

Die Hauptteile der Hardware, nämlich Eingangstastatur, Mikroprozessor, eine Reihe von Speichern verschiedener Art, Drucker und Sichtgerät müssen zu funktionierenden Einheiten zusammengeschlossen werden. Die Anleitung hierzu findet sich in diesem Buch in allen Einzelheiten und vor allem in der Auswahl preiswerter, aber doch moderner, auch kommerziell üblicher Teile.

Solche Geräte und Anlagen ermöglichen auch die Steuerung und Bedienung von Meß- und Anzeigegeräten. Der Selbstbau eines solchen Peripheriegerätes ist ebenfalls ausführlich beschrieben und sein Anschluß dargestellt.

Einen sehr breiten Raum nimmt die Anwendung der selbst erstellten Anlage ein. Hierfür sind eine Reihe ausführlicher Programme erarbeitet, die zahlreiche Spiele, Lösungen mathematischer Aufgaben und wissenschaftlicher Probleme zu bearbeiten gestatten.

Schließlich enthält das Buch eine Reihe von Anregungen, um selbst Programme aufzustellen und sie in den gebauten Geräten zu erproben.

Einem großen Kreis von Interessenten soll das Werk Anregungen geben für Hobby und Studium, für Lernen und Erfinden, in einer neuen, hochinteressanten Technik, die der Elektronik Eingang in Gebiete verschafft, von denen wir heute nur träumen können. Hier handelt es sich um eine Technik, die in nächster Zeit neue Aufgaben schafft, die rationalisieren und den Lebensstandard erhöhen hilft, die mit Sicherheit denjenigen, die lernen etwas davon zu verstehen, den zukünftigen Arbeitsplatz sichert.

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen und technischen Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag sieht sich deshalb gezwungen, darauf hinzuweisen, daß er weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen kann. Für die Mitteilung eventueller Fehler sind Autor und Verlag jederzeit dankbar.

Inhalt



1 Aufbau-Prinzip

In den Kapiteln 2 bis einschließlich 8 sollen verschiedene Geräte beschrieben werden. Es ist möglich, diese Geräte unterschiedlich zu verknüpfen.

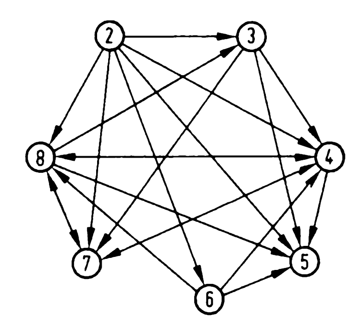


Abb. 1-1 Verknüpfungsmöglichkeit der Geräte

Abb. 1-1 zeigt ein Schema, das angibt, wie sich die Geräte der einzelnen Kapitel sinnvoll verknüpfen lassen. Ferner ist durch einen eingezeichneten Pfeil die Richtung des Datenflusses angegeben. Ist in beiden Richtungen ein Pfeil eingezeichnet, so heißt das, daß der Datenfluß in beiden Richtungen möglich ist.

Beispielsweise führen von Kapitel 2 Pfeile zu allen anderen Kapiteln (3...8). Kapitel 2 beschreibt nämlich die Stromversorgung. Durch die Darstellung wird verdeutlicht, daß die beschriebene Stromversorgung für alle Geräte der anderen Kapitel verwendbar ist.

2 Grundausstattung

In diesem Kapitel sollen drei Geräte beschrieben werden, die zum Aufbau der folgenden Schaltungen notwendig bzw. nützlich sind.

2.1 Stromversorgung

Die Schaltungen arbeiten vorwiegend in integrierter Technik mit TTL- und MOS-ICs.

Die TTL-ICs (74er Serie) benötigen eine Versorgungsspannung von 5 V ± 5 %.

Manche der hier verwendeten MOS-ICs brauchen noch eine zusätzliche Versorgungsspannung von 12 V.

Bei der Wahl einer geeigneten Stabilisierschaltung halfen außerdem folgende Kriterien:

1. Die verwendete Stabilisierschaltung muß weitgehend kurzschlußsicher sein.
2. Die Schaltung muß betriebssicher sein.
3. Eine gute Ausregelung von Laständerungen und Schwankungen der Versorgungsspannung muß gewährleistet sein.

Als maximaler Ausgangsstrom einer 5-V-Stabilisierschaltung wurde 2 A gewählt. Dieser Ausgangsstrom ruft schaltungstechnisch keine Schwierigkeiten hervor und ist für die meisten Einzelschaltungen ausreichend. Bei größeren Schaltungskomplexen verwendet man dann mehrere solcher Einheiten. Diese Methode kommt auch sehr der Störsicherheit des ganzen Systems zugute.

Für die 12-V-Stabilisierungseinheiten wurde ebenfalls ein maximaler Strom von 2 A gewählt. Diese Einheit kann somit einen für das gesamte Schaltungssystem ausreichenden Strom liefern.

Die Schaltung

Als Spannungsregler wurde ein integrierter Schaltkreis verwendet. Dieses IC enthält eine Referenzspannungsquelle und einen Operationsverstärker. Durch die Art der äußeren Beschaltung kann der einstellbare Ausgangsspannungsbereich festgelegt werden.

Abb. 2.1-1 zeigt die Beschaltung für die 5-V-Stabilisierung. Da das IC von sich aus nur maximal 150 mA liefern kann, wurden noch die Transistoren T 1 und T 2 hinzugefügt, so daß der höchst entnehmbare Ausgangsstrom nun mit Sicherheit 2 A betragen kann. Mit P 1 wird die Ausgangsspannung auf 5 V abgeglichen. Mit P 2 stellt man den Einsatz der Strombegrenzung ein. Der Abgleich des Trimmers P 2 geschieht folgendermaßen. Man belastet den Ausgang mit einem Widerstand

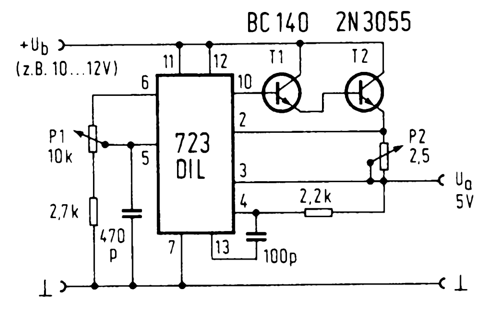


Abb. 2.1-1 Stabilisierungsschaltung für 5 V

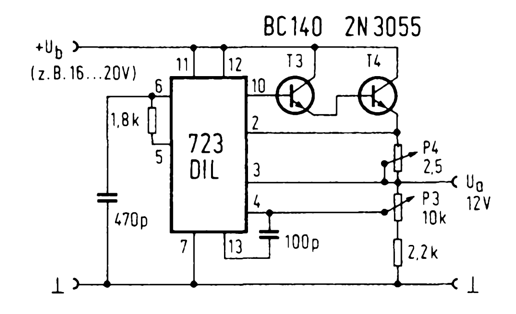


Abb. 2.1-2 Stabilisierungsschaltung für 12 V

von 2,5 Ω und regelt den Wert von P 2 so ein, daß an einem angeschlossenen Spannungsmesser gerade noch 5 V gemessen werden. Somit würde bei einer größeren Last (oder gar Kurzschluß) die Strombegrenzung einsetzen und die Schaltung vor Überlastung schützen.

In *Abb. 2.1-2* sieht man die Beschaltung des ICs, wie sie für die Erzeugung der 12-V-Spannung notwendig ist. Die Ausgangsspannung wird mit P 3 auf 12 V eingestellt und die Strombegrenzung mit P 4.

Nachstehende *Tabelle 2.1-1* zeigt die Bezeichnung der Anschlüsse für DIL- und TO-Gehäuse des ICs Typ 723 für die Spannungsregelung.

Abb. 2.1-3 zeigt eine mögliche Schaltung für eine vollständige Stromversorgung.

Tabelle 2.1-1

DIL	TO
11	8
12	7
10	6
2	10
3	1
4	2
13	9
7	5
5	3
6	4

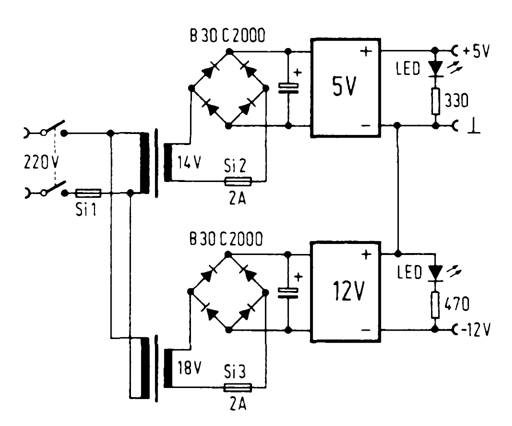


Abb. 2.1-3 Beispiel eines Netzgerätes

2.2 Prüfstift für TTL-Schaltungen

Da in umfangreichen Schaltungen immer wieder Fehler vorkommen, sei es bei einem Versuchsaufbau oder beim Bestücken einer gedruckten Schaltung, ist es notwendig, mit einem geeigneten Prüfinstrument die an den einzelnen Bauelementen vorkommenden Signale festzustellen, um so den eventuell vorhandenen Fehler lokalisieren zu können.

In digitalen Schaltungen mit TTL-ICs ergeben sich aber für die Signale nicht nur zwei sondern drei Möglichkeiten. Das auf einer Leitung befindliche Signal kann das Zeichen „ein“ oder das Zeichen „aus“ bedeuten, die Leitung oder der Anschluß können sich aber auch in einem undefinierten Zustand befinden. Bei einer Unterbrechung wird das Zeichen „aus“ vorgetäuscht. Alle drei Zustände sollte man mit dem Prüfinstrument unterscheiden können.

Die im folgenden beschriebene Schaltung soll mit einem Prüfstift eine Anzeige erlauben, aus der hervorgeht, ob an dem Anschluß ein Signal 1, z. B. High, oder 0, dann z. B. Low bedeutet, oder eine Wechselspannung, die abwechselnd die beiden Zustände 1 und 0 annimmt. Es ist aber auch möglich, daß der Anschluß unbeschaltet ist bzw. daß der Ausgang

eines ICs sich im Zustand „open collector“ befindet.

Alle diese Erkenntnisse sind bei Anwendung des beschriebenen Prüfstiftes möglich, obwohl dieser nur mit drei Transistoren ausgerüstet ist.

Die Schaltung

Die eben beschriebenen Eigenschaften werden mit Hilfe von zwei Lumineszenz-Dioden als Anzeige erreicht. Dabei bedeuten die Anzeigen der beiden mit H und L bezeichneten LEDs folgendes:

1. Liegt am Eingang des Prüfstiftes ein Signal H mit dem Zustand 1, so leuchtet nur die Lumineszenz-Diode H.

2. Liegt am Eingang des Prüfstiftes ein Signal L mit dem Zustand 0, so leuchtet nur die Lumineszenz-Diode L. Da in diesem Anzeigezustand aber auch eine Wechselspannung anliegen kann, die von einem offenen Kollektor ausgeht, muß eine Unterscheidungsmöglichkeit geboten werden. Dazu ist die Taste Ta 1 vorgesehen, die auch in *Abb. 2.2-1* eingezeichnet ist, welche die gesamte Schaltung des Prüfstiftes zeigt. Wird die Taste betätigt, dann leuchtet entweder weiterhin nur die LED L, oder aber es leuchten beide LEDs auf. Im ersten Fall handelt es sich mit größter Wahrscheinlichkeit um das Signal L, im anderen Fall ist eine Wechselspannung vorhanden.

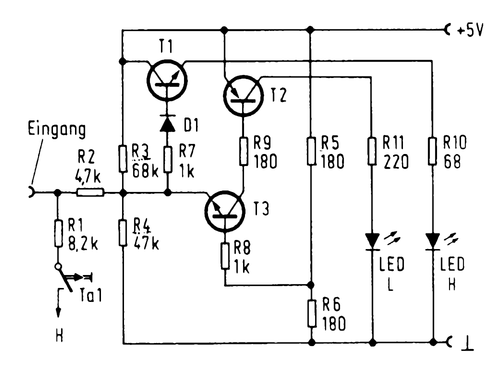


Abb. 2.2-1 Schaltung des Prüfstiftes

3. Wenn nach dem Berühren der Meßstelle keine der Lumineszenz-Dioden aufleuchtet, so ist der Anschluß, wenn es sich um einen Eingang handelt, unbeschaltet. Wenn keine der LEDs leuchtet und man an einem Ausgang prüft, so handelt es sich sehr wahrscheinlich um einen unbeschalteten „open collector“ Ausgang.

Die Wirkung der Schaltung, die die Abb. 2.2-1 zeigt, beruht im wesentlichen auf einer Brückenschaltung, die aus den vier Widerständen R 3, R 4, R 5 und R 6 besteht. Die Speisung der Brücke erfolgt mit 5 V. Die Brücke ist im Ruhezustand wegen der Verschiedenheit der Widerstände nicht abgeglichen. In der Diagonale liegt mit Basis und Emitter der Transistor T 3, dessen Kollektor zur Basis des Transistors T 2 führt, in dessen Kollektorkreis die Lumineszenz-Diode L liegt.

Der Emitter des Transistors T 3 ist über einen Widerstand R 7 und eine Diode D 1 mit der Basis des dritten Transistors T 1 verbunden, in dessen Emitterkreis über einen Widerstand R 10 die Lumineszenz-Diode H liegt. Der Eingang des Prüfstiftes führt über einen Widerstand R 2 zum Emitter des in der Brükkendiagonale liegenden Transistors T 3.

Der Eingang des Prüfstiftes ist ferner über einen Widerstand R 1 und eine Taste Ta 1 zu einem Signal H geführt, so daß das Potential im Falle eines offenen Kollektors vor dem Widerstand R 2 zu H (etwa +5 V) definiert werden kann [1].

Wirkungsweise

Wenn man ein Signal 0 (L) an den Eingang legt, so wird der Emitter des Transistors T 3 negativ gegenüber der Basis, die zwischen den beiden Widerständen R 5 und R 6 über den Widerstand R 8 an 2,5 V liegt. Der Transistor T 3 wird leitend. Dadurch wird der Transistor T 2 ebenfalls leitend. Die Lumineszenz-Diode LED L leuchtet auf. Der Transistor T 1 sperrt, da dessen Basis das entsprechende Potential hat.

Legt man den Eingang des Prüfstiftes an ein Signal 1 (H), so wird der Transistor T 1 leitend. Dadurch erfolgt eine Anzeige mit der Lumineszenz-Diode H. Der Transistor T 3 sperrt, da dessen Emitter gegenüber der Basis positiv geworden ist.

Wenn der Eingang der Prüfeinrichtung dadurch, daß kein Potential anliegt, völlig offen ist, so kann weder der Transistor T 1 noch der Transistor T 3 leitend werden, da für keinen von beiden das richtige Potential vorhanden ist.

Mit der Taste Ta 1 ist es, wie schon erwähnt, möglich, ein Signal H an den Eingang zu legen. Dadurch wird erreicht, einem offenen Kollektor eine definierte Spannung zu geben, so daß man durch die Anzeige zwischen Signal 0 (L) und einer etwa anliegenden Wechselspannung unterscheiden kann.

Als Transistor für T 1 und T 3 kann man jeden npn-Transistor einsetzen (z. B. BC 107). Für den Transistor T 2 empfiehlt sich ein pnp-Silizium-Transistor mit $h_{FE} > 100$ (z. B. BC 177). Für die Diode verwendet man am besten eine Ausführung in Germanium.

2.3 Pegelanzeige für die Anschlüsse von TTL-Bausteinen auf dem Oszilloskop

Wenn ein digitales IC in eine Schaltung eingebaut ist, so ist es oftmals zur Fehlersuche wünschenswert, die an den Anschlüssen befindlichen Pegel gleichzeitig beobachten zu können. Dabei sollen nicht nur statische, sondern auch schnell wechselnde Pegel erkannt werden [2].

Mit Hilfe eines handelsüblichen Testclips, das sich direkt auf das IC stecken läßt, können alle Anschlüsse bequem herausgeführt werden.

Für die Anzeige der Pegel wurde der Bildschirm eines Oszilloskops gewählt, wobei sowohl der Y- als auch der X-Eingang gebraucht werden.

Die Darstellung der Pegel erfolgt entsprechend der Anordnung der Anschlüsse des ICs

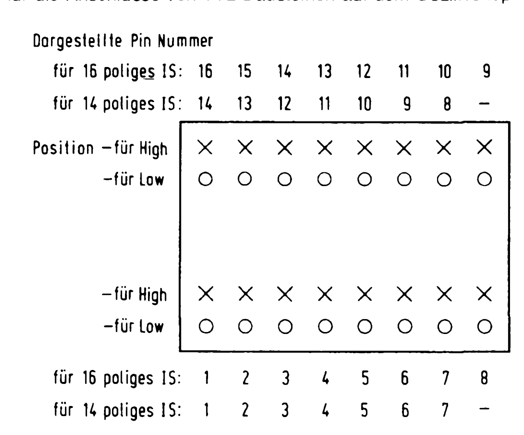


Abb. 2.3-1 Mögliche Positionen des Elektronenstrahls

in zwei Reihen mit je einer Position für die Pegel L und H entsprechend *Abb. 2.3-1*.

Auf dem Bildschirm erscheinen 16 Leuchtpunkte, und zwar die in Abb. 2.3-1 kreisförmig dargestellten, wenn alle Potentiale auf L (Low) liegen. Diejenigen Pegel, die H (HIGH) entsprechen, erscheinen an den mit Kreuzen bezeichneten Positionen nach oben versetzt. Bei einem Potentialwechsel geringer Frequenz ist der Sprung von H auf L und umgekehrt deutlich erkennbar. Bei höheren Frequenzen erscheinen beide Positionen scheinbar gleichzeitig als zwei übereinander liegende Leuchtpunkte.

Prinzipieller Funktionsablauf der Schaltung

In *Abb. 2.3-2* ist ein Flußdiagramm dargestellt, das den Funktionsablauf der Schaltung angibt.

Bei Betrachtung des Flußdiagramms fällt auf, daß dieses keinen Eingang und keinen Ausgang besitzt, daß also kein Start- und kein Stoppbefehl vorkommt. Dies entspricht der Schaltung, denn nach dem Einschalten dieses Gerätes sind die Zustände der einzelnen in der Schaltung vorkommenden Flipflops nicht definiert. Bei einer solchen Schaltung muß also gewährleistet sein, daß die zunächst nicht definierten Zustände der Flipflops schon nach einer kurzen Zeit in definierte übergehen.

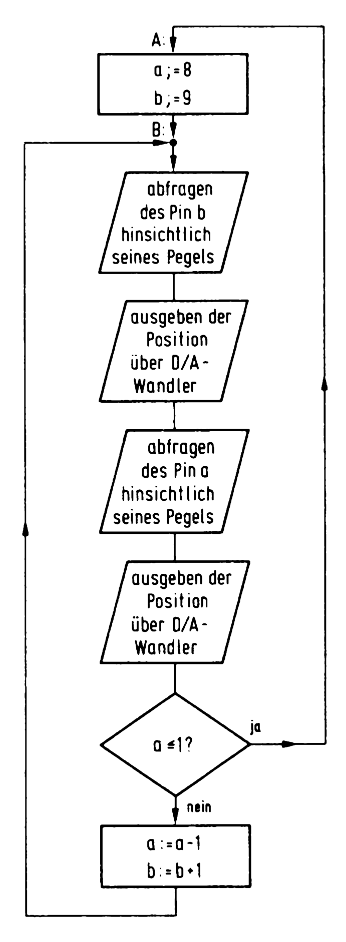


Abb. 2.3-2 Ablaufschema

Diese Bedingung ist sowohl in der Schaltung als natürlich auch in dem dazugehörigen Flußdiagramm erfüllt. Zur Erklärung des Flußdiagramms beginnen wir der Einfachheit halber bei Marke A. Der Begriff „Marke" bezeichnet dabei eine Stelle in einem Programm, die eine Adresse besitzt und mit einer Kennzeichnung versehen sein kann. Somit kann ein einfacher Bezug auf diese Stelle durch Verwendung des mit „Marke" bezeichneten Begriffs hergestellt werden. Es werden zwei Variable (a und b) verwendet, die als Zähler für die jeweilige Pinnummer verwendet werden. Dabei wird die Variable a als Zähler für die Pins 1...8 verwendet und die Variable b als Zähler für die Pins 9...16.

Wenn man also bei der Marke A in das Flußdiagramm eintritt, so werden zunächst die beiden Variablen a und b mit definierten Werten besetzt. Die Variable a erhält dabei den Wert 8 zugewiesen und die Variable b erhält den Wert 9 zugewiesen. (Der Wert 8 ist der höchste Wert, der bei der Variablen a vorkommen kann, und der Wert 9 der kleinste Wert für die Variable b.)

Als nächstes wird das Pin mit der Nummer (Inhalt von b)* auf seinen Zustand abgefragt. Dann wird auf dem Oszilloskop die Position für den Strahl ausgegeben. Diese ist bestimmt durch den Inhalt von b und zusätzlich den Pegel, der an dem abgefragten Pin liegt.

Das gleiche geschieht nun für den Pin mit der Nummer (Inhalt von a)*. Es erfolgt nun eine Abfrage nach dem Wert der Variablen a. Ist der Wert kleiner oder gleich 1, so erfolgt ein Sprung zur Marke A, das heißt, den Variablen wird erneut ein definierter Wert zugewiesen.

Ist a aber größer als 1, so wird a um eins erniedrigt und b um eins erhöht (in der Sprache der Informatik ausgedrückt: Der Inhalt von a wird um eins decrementiert und der Inhalt von b um eins incrementiert). Damit werden die beiden nächsten Nummern für die abzufragenden Pins bestimmt. Es erfolgt dann ein Sprung zur Marke B. Dort wiederholt sich die Abfrage.

Die Schaltung

Abb. 2.3-3 zeigt die praktische Ausführung der Schaltung.

Die 16 Anschlüsse des Testclips (Testclip selbst nicht eingezeichnet) werden zwei Multiplexer M 1 und M 2, mit je 8 Anschlüssen zugeführt.

---

* Nummer (...) bedeutet eine durch den Inhalt der Klammer bezeichnete Nummer.

Das Steuern des Ablaufs wird von einem von außen zugeführten Takt erreicht. Dieser Takt C 1 gelangt an ein Flipflop FF 1, das abwechselnd die beiden Multiplexer M 1 und M 2 steuert, so daß immer nur einer davon in Betrieb ist. Gleichzeitig wird mit der abfallenden Flanke des $\bar{Q}$-Ausgangs dieses Flipflops ein als 3-bit-Dual-Zähler geschaltetes IC SN 7490 um eins fortgeschaltet.

Die Ausgangspegel der Ausgänge dieses Zählers bestimmen die Nummer des Pins, das auf seinen Zustand hin abgefragt werden soll. Durch die Steuerung der beiden Multiplexer durch den Zähler und das Flipflop wird also ermöglicht, daß die 16 Eingänge beider Multiplexer nacheinander abgefragt werden. Die beiden Ausgänge der beiden Multiplexer sind über die zwei Widerstände R 1 und R 2 miteinander verbunden und führen zum Y-Ausgang. Dadurch wird erreicht, daß die Information über die Pegel an den abgefragten Pins in ein entsprechendes Analogsignal umgewandelt wird. Über den Widerstand R 3 wird noch das Signal des Q-Ausgangs des Flipflops auf den Y-Ausgang gemischt. Damit wird auch noch die Information, welche der beiden Pinreihen gerade abgefragt wird, in eine analoge Form übergeführt. Wenn man dieses Signal an den Y-Eingang eines Oszilloskops legt, ohne den X-Eingang zu beschalten, so erhält man je nach anliegenden Signalen zwei bis vier übereinanderliegende Punkte.

Die Aufgabe der Erzeugung eines Signals für den X-Eingang übernimmt ein Digital-Analogumsetzer, der aus den Verknüpfungen N 1...N 3 und den Widerständen R 4...R 9 aufgebaut ist. Als Eingangswerte erhält er die Signale des Zählers. An dem Ausgang X steht dann eine Treppenspannung an. Wenn man nun auch diesen Ausgang an den Eingang des Oszilloskops anschließt, so ergibt sich auf dem Bildschirm das gewünschte Bild mit der Information über die an den Anschlüssen des Testklips anliegenden Signale.

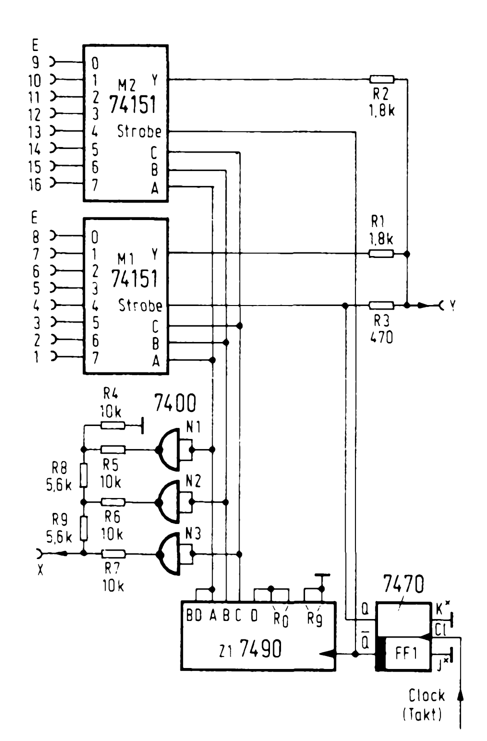


Abb. 2.3-3 Schaltung des Prüfgerätes

Eigenschaften des Gerätes

Die Frequenz des Taktes sollte veränderbar sein, damit die Abtastung nicht synchron zu einem an den Anschlüssen des zu prüfenden ICs vorhandenen Takt ist. Dadurch könnte sich aufgrund von Interferenzerscheinungen ein falsches Bild ergeben. Die Schaltung arbeitet einwandfrei mit Taktfrequenzen von 600 Hz bis 10 kHz. Unterhalb von 600 Hz würde das Bild durch die Frequenzuntersetzung zu stark flimmern.

An den Anschlüssen des Testklips können Spannungen mit Frequenzen von 0 Hz bis 10 MHz liegen. Bei den höchsten Frequenzen ist an Stelle zweier Punkte für L und H ein Strich zu erkennen, so daß auch hier ein dauernder Pegelwechsel erkennbar bleibt.

Der in dieser Schaltung verwendete Testclip, mit vergoldeten Kontakten, ist von der Firma Fischer Elektronik lieferbar.

3 Universalzähler

Für den Aufbau von Universalzählern gibt es sehr viele Schaltungsbeschreibungen, sei es in Zeitschriften oder Büchern. Hier soll aber nicht ein herkömmlicher Universalzähler beschrieben werden, sondern ein Universalzähler mit einer Mikroprogrammsteuereinheit. Worum es sich dabei genau handelt, und wie man damit arbeiten kann, soll in diesem Kapitel näher beschrieben werden.

Abb. 3-1 zeigt das Blockschaltbild des Gerätes. Man erkennt darauf fünf verschiedene Einheiten: Den Anzeigeteil, der die Aufgabe hat, die anfallenden Daten dem Benutzer mitzuteilen, den Eingangsteil, der die Aufgabe hat, Bedienungsinformationen und Eingangssignale, wie Frequenz, aufzunehmen und dem Verarbeitungsteil weiterzuleiten. Dann gibt es natürlich noch den Taktgenerator, der das genaue notwendige Taktsignal für den Verarbeitungsteil (Mikroprogrammsteuereinheit) liefert, ferner die Mikroprogrammsteuereinheit, die die Aufgabe hat, sämtliche Steuer- und Überwachungsfunktionen zu übernehmen und die Daten zwischenzuspeichern. Schließlich ist noch der Programmspeicher vorhanden, der die Information über den gesamten Funktionsablauf beinhaltet.

3.1 Anzeigeteil

Abb. 3.1-1 zeigt die vollständige Schaltung des Anzeigeteils. Über die Ausgänge A.B.C.D. des Anzeigeteils gelangt die Information darüber, welche Stelle gerade angezeigt werden soll, an die Mikroprogrammsteuereinheit. Die Information, welche Zahl gerade an dieser Stelle angezeigt werden soll, gelangt über die Anschlüsse AI, BI, CI, DI im BCD-Code an

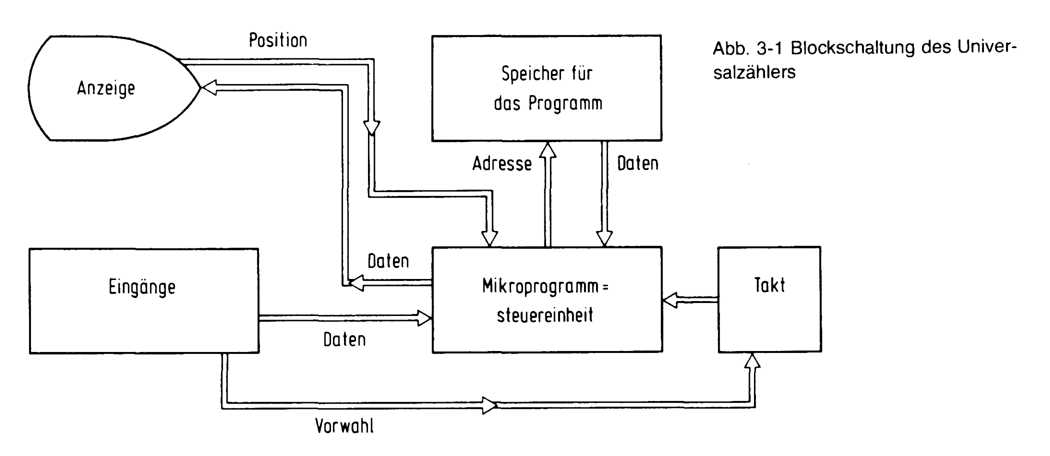
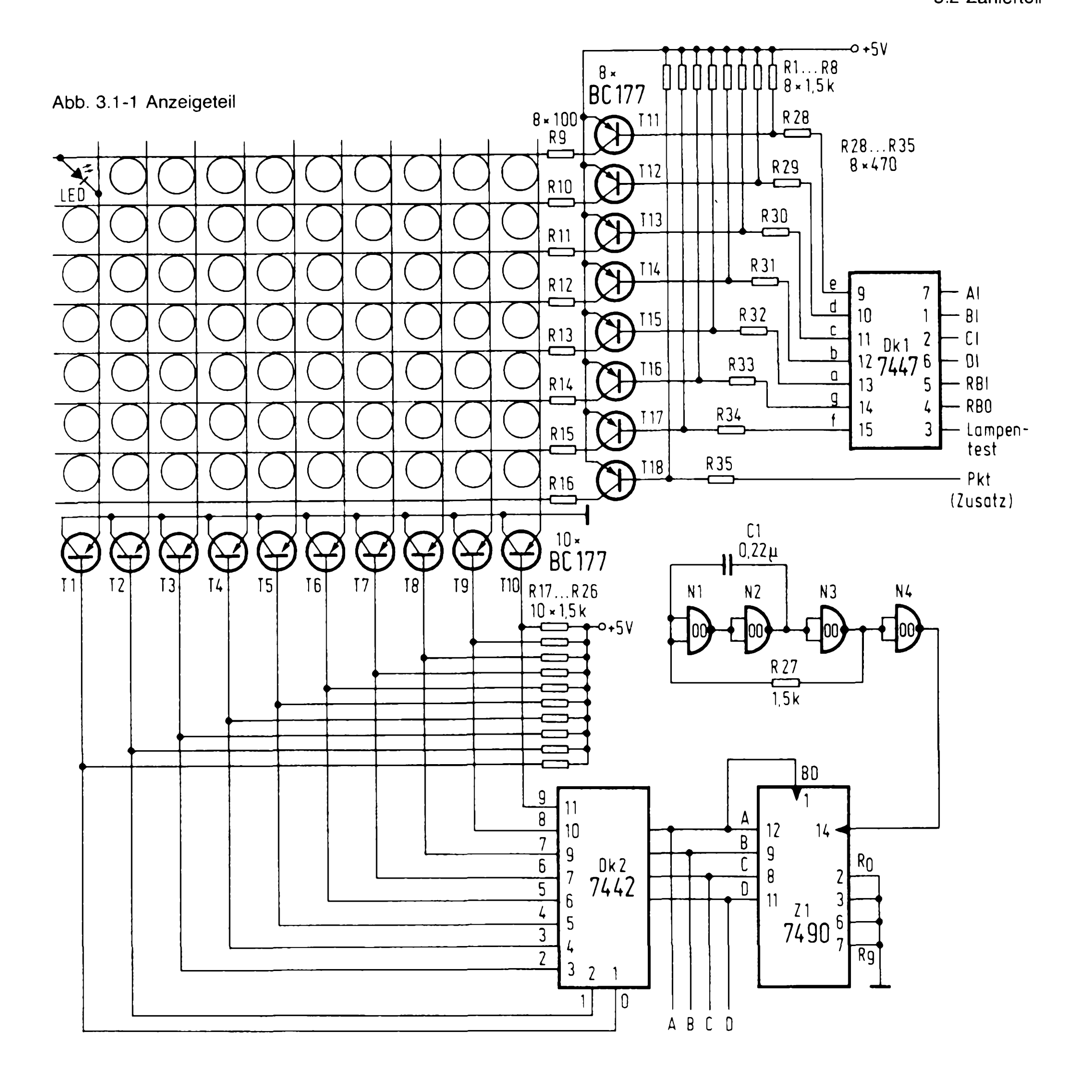


Abb. 3-1 Blockschaltung des Universalzählers

Abb. 3.1-1 Anzeigeteil



den Anzeigeteil. Dort wird die Information umcodiert und so für die Siebensegmentanzeigen aufbereitet. Über Treiberstufen (T 11...T 18) gelangen die Signale an die Siebensegmentanzeigen. Mit Hilfe eines Taktgenerators wird ein als Dezimalzähler geschaltetes IC SN 7490 angesteuert. Dieser Zähler bestimmt die Stelle, die gerade angezeigt werden soll. Die Information gelangt über einen Umcodierer, der das BCD-Signal (8-4-2-1) in den (1-aus-10-)Code umwandelt, und über Treiberstufen (T 1...T 10) ebenfalls an die Siebensegmentanzeigen. Die Taktfrequenz des Generators (N 1...N 4, C 1, R 27) wurde so gewählt, daß sich ein flimmerfreier Eindruck ergibt.

3.2 Zählerteil

Der Zählerteil hat die Aufgabe, die eigentliche Messung durchzuführen und den sich ergebenden Wert kurz zwischenzuspeichern. *Abb.*

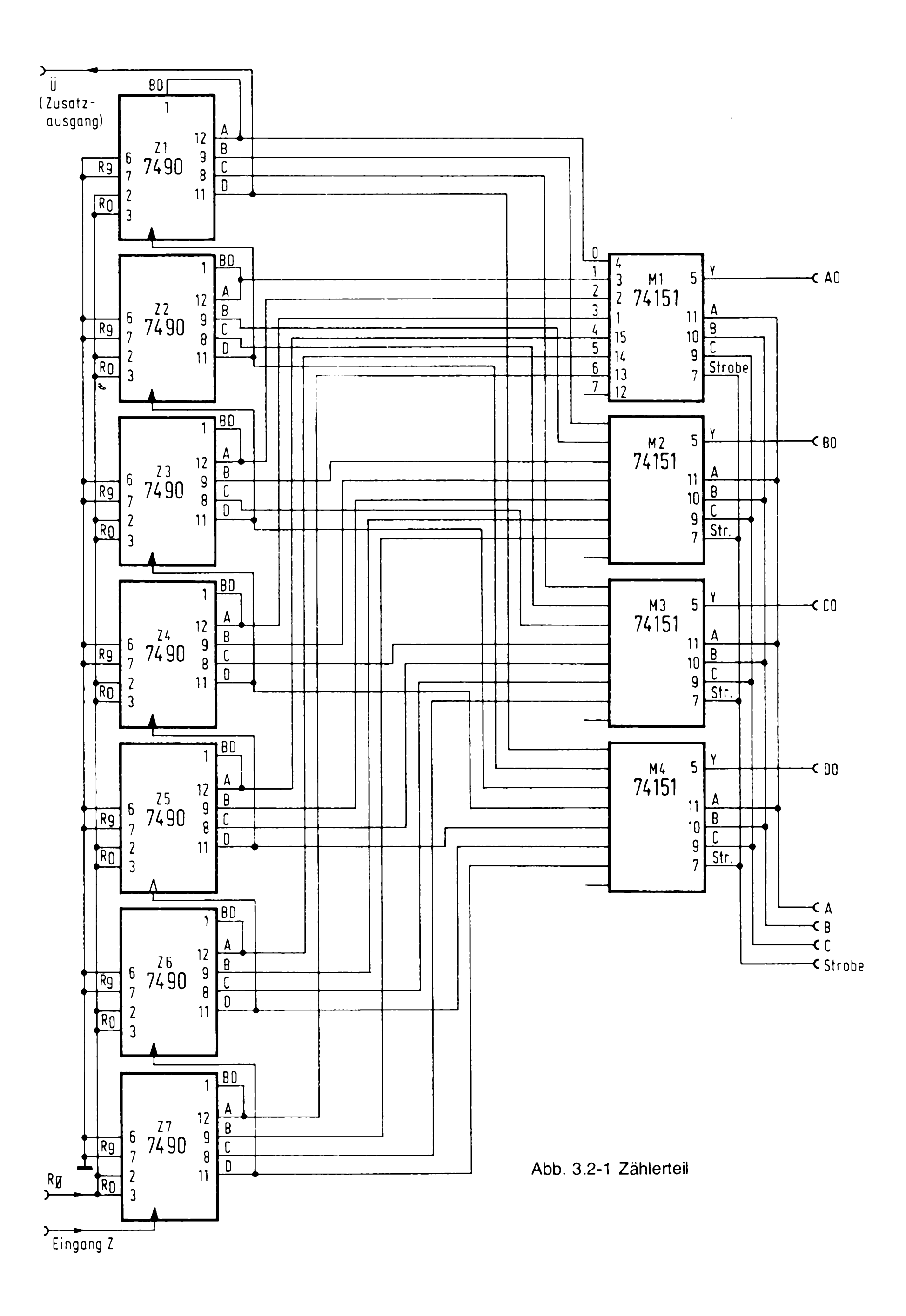


Abb. 3.2-1 Zählerteil

3.2-1 zeigt die Schaltung dieses Teils. Der Zählerteil besteht insgesamt aus den als Dezimalzähler geschalteten ICs SN 7490 (Z 1...Z 7) und vier Multiplexern des Typs SN 74151 (M 1...M 4). Die Zähler werden durch zwei Eingangssignale gesteuert. Der eine Eingang (R 0) dient der Rücksetzung des Zählerinhalts und der andere Eingang (Z) dient der Erhöhung des Inhalts dieser Zählerkette um die Anzahl der zugeführten Impulse an diesem Eingang. Die an den Zählerausgängen anstehenden Daten über den Zählerstand werden mit Hilfe eines Multiplexerteils für die Programmsteuereinheit erreichbar gemacht. Den Multiplexern (M 1...M 4) des Typs SN 74151 wird über drei Eingänge die Information erteilt, welcher der Zähler (Z 1...Z 7) gerade abgefragt werden soll und über den vierten Eingang wird bestimmt, ob die Multiplexer überhaupt Information abgeben sollen. An den vier Ausgängen der Multiplexer (M 1...M 4) liegt die jeweilige Dateninformation für die Mikroprogrammsteuereinheit.

3.3 Quarz-Taktgeber mit Frequenzteiler (Zeitbasis)

Er dient, wie schon gesagt, dazu, die ganzen Zeitabläufe der gesamten Schaltung zu bestimmen, also das sogenannte „timing" zu übernehmen.

Abb. 3.3-1 zeigt die Schaltung. Der eigentliche Quarzgenerator besteht aus den NAND-Verknüpfungen N 1 und N 2, ferner aus den Bauelementen C 1, R 1, R 2, Q 1, C 2 und C 3. Mit C 3 kann ein genauer Abgleich auf die Frequenz 1 000 000 Hz durchgeführt werden. Das Ausgangssignal gelangt nun über zwei weitere NAND-Verknüpfungen an die Teilerkette. Diese besteht aus den Zählern Z 1...Z 3, die ebenfalls als Dezimalzähler geschaltet sind. Es wird jeweils eine „Teilung" durch 10 erreicht. Mit den Schaltern SA, SB, SC kann die die Schaltung steuernde Taktfrequenz ausgewählt werden. Dadurch wird in der Betriebsart „Frequenzmeßgerät" die Umschaltung des Meßbereichs erzielt. In der Stellung 0,1x, d. h. wenn Schalter SA betätigt ist,

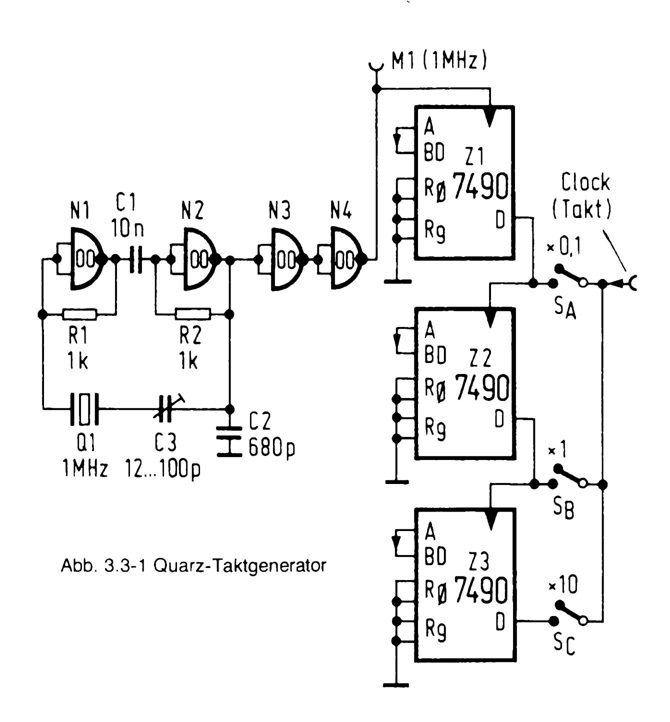


Abb. 3.3-1 Quarz-Taktgenerator

ergibt sich folgender Bereich: XXXXX.XX kHz. Wenn SB betätigt ist: XXXX.XXX kHz und bei Betätigung von SC XXX.XXXX kHz.

3.4 Mikroprogrammsteuereinheit

In diesem Abschnitt wird nun der Kern des Universalzählers behandelt.

Es soll zunächst einmal zum besseren Verständnis die prinzipielle Entwicklung und der Aufbau einer Mikroprogrammsteuereinheit erklärt werden.

Abb. 3.4-1 zeigt das Blockschaltbild einer einfachen Mikroprogrammsteuereinheit. Diese Mikroprogrammsteuerung besteht aus drei verschiedenen Teilen.

Der Programmzähler bestimmt die Adresse des gerade auszuführenden Befehls.

Der ROM (read only memory = Nur-Lese-Speicher) speichert die Befehle unter einer bestimmten Adresse.

Der Decodier- und Treiberteil entschlüsselt die Befehle und gibt sie weiter.

Die angegebene Schaltung ist dann in der Lage, sogenannte lineare Programme abzuarbeiten.

Doch was versteht man unter einem Programm?

Ein Programm ist eine Abfolge von Anweisungen.

Zum Beispiel:

1. Anweisung: Zähler x auf Null setzen
2. Anweisung: Zählereingang freigeben
3. Anweisung: Zählereingang schließen
4. Anweisung: Zählerstand ausgeben.

Es wäre dann noch der Begriff „lineares Programm“ zu klären. Unter einem linearen Programm versteht man ein Programm, das die Anweisungen der Reihenfolge im Speicher (genauer gesagt der aufsteigenden Folge ihrer Adressen) entsprechend abarbeitet, das heißt ausführt. Bei dem angeführten Beispiel zur Erklärung des Begriffs Programm handelt es sich sogar um ein solches lineares Programm, da nirgends innerhalb des Programms die Anweisung gegeben wurde, die Abarbeitungsreihenfolge zu ändern.

In der in Abb. 3.4-1 dargestellten Blockschaltung erkennt man die Unfähigkeit dieser Schaltung nichtlineare Programme abarbeiten zu können. Es fehlt eine Rückführung zum Programmzähler, die diesen befähigen könnte, die Reihenfolge (die Folge der Adressen) der Abarbeitung der Anweisungen zu ändern.

In *Abb. 3.4-2* ist die Blockschaltung einer universellen Mikroprogrammsteuerung dargestellt, die auch die Fähigkeit besitzt, nichtlineare Programme auszuführen.

Beispiel eines nichtlinearen Programms:

1. Anweisung: Setze den Zähler x auf Null
2. Anweisung: Erhöhe den Zählerinhalt um eins
3. Anweisung: Gib den Zählerstand aus
4. Anweisung: Spring zurück nach Anweisung 2
5. Anweisung: Leer.

Dieses Programm enthält eine solche Sprunganweisung. Sie bewirkt, daß nach Ausführung der 4. Anweisung nicht die 5. Anweisung, sondern die 2. Anweisung ausgeführt wird. Da-

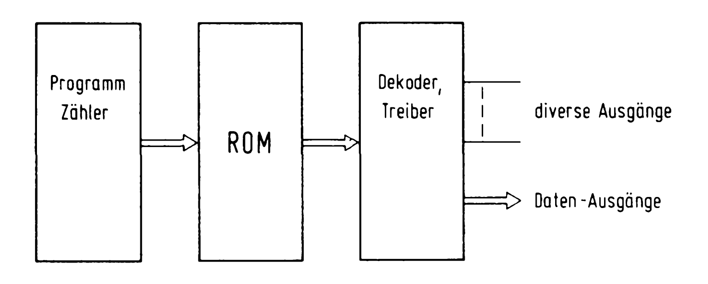


Abb. 3.4-1 Mikroprogrammsteuereinheit für lineare Programme

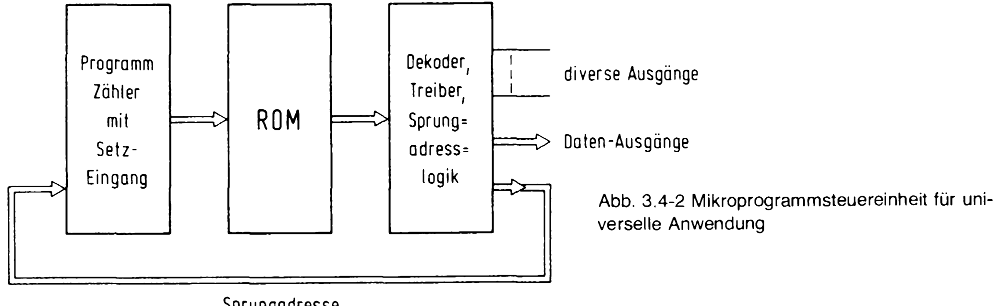


Abb. 3.4-2 Mikroprogrammsteuereinheit für universelle Anwendung

durch wird erzielt, daß sich die Ausführungen der Anweisungen 2...4 zyklisch wiederholen. Es handelt sich bei der 4. Anweisung um einen sogenannten unbedingten Sprung. Dieser Sprungbefehl wird in jedem Fall ausgeführt. Es gibt aber auch noch sogenannte bedingte Sprungbefehle, die in Abhängigkeit von bestimmten Zuständen ausgeführt oder nicht ausgeführt werden.
Beispiel für ein Programm mit bedingten Sprunganweisungen:

1. Anweisung: Spring nach Anweisung 1, wenn die Starttaste nicht betätigt wurde.*
2. Anweisung: Setz den Zähler x auf den Wert 0.
3. Anweisung: Erhöhe den Zählerinhalt um eins.
4. Anweisung: Spring nach Anweisung 3, wenn die Stopptaste nicht betätigt wurde.
5. Anweisung: Gib den Zählerstand aus.
6. Anweisung: Spring nach Anweisung 1.

Das Beispiel stellt das Programm für eine Stoppuhr dar.

Erklärung des Programms:

Wenn bei der Ausführung der ersten Anweisung die Starttaste nicht betätigt wurde, so erfolgt ein Sprung zur ersten Anweisung und damit ein erneutes Ausführen der ersten Anweisung. Diese Anweisung bewirkt ein Warteverhalten. Es wird solange gewartet, bis die Taste „Start" betätigt wird. Dann folgt das Ausführen der zweiten Anweisung. Dabei wird nun der Inhalt eines Zählers x auf 0 gesetzt. Nun folgt die Ausführung der dritten Anweisung. Der Zählerinhalt wird um eins erhöht. Bei der vierten Anweisung liegt wieder ein bedingter Sprung vor. Wurde die Stopptaste nicht betätigt, so folgt ein Sprung zur dritten Anweisung und damit eine Wiederholung des Programmteils von da an. Wurde aber die Stopptaste betätigt, so erfolgt bei der Ausführung der vierten Anweisung kein Sprung und es wird die fünfte Anweisung ausgeführt. Sie beinhaltet den Befehl „Gib den Zählerstand aus". Es erfolgt nun also die Ausgabe des Zählerstands und somit erfolgt eine Anzeige der gestoppten Zeit. Die sechste Anweisung enthält einen unbedingten Sprungbefehl zur Anweisung 1, womit eine erneute Wartestellung eingeleitet wird, und das ganze von vorne beginnt.

Für das ordnungsgemäße Funktionieren dieses Programms ist es notwendig, nach Betätigen einer der beiden Tasten, deren Schaltzustand zu speichern. Das erfolgt solange, bis die dazugehörige Sprunganweisung ausgeführt wurde, um so eine Abfrage der Sprunganweisung zu ermöglichen, ob die Taste betätigt wurde oder nicht.

Nach dieser Exkursion in die theoretischen Grundlagen einer Programmsteuerung ist es

* Erklärung folgt

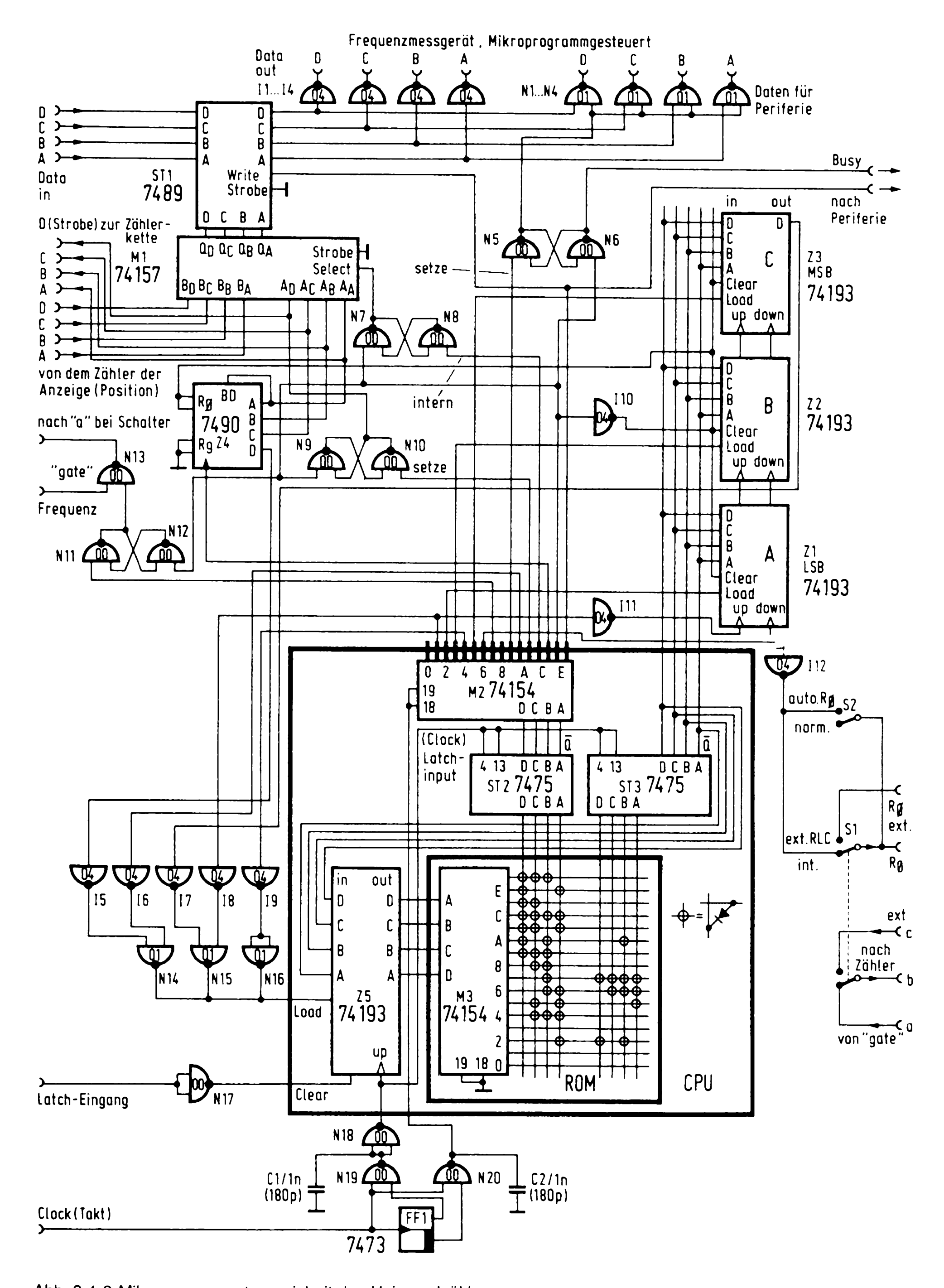


Abb. 3.4-3 Mikroprogrammsteuereinheit des Universalzählers

nun möglich, auf die genaue Schaltung der Mikroprogrammsteuerung einzugehen. *Abb. 3.4-3* zeigt die gesamte Schaltung der Mikroprogrammsteuerung, sowie einige zusätzliche Elemente, die für die Anwendung in dem Universalzähler nötig sind.

Der innerste stark umrandete Teil der Schaltung zeigt den Programmspeicher. Dieser ist als ROM (Nur-Lese-Speicher) ausgeführt, da das Programm im allgemeinen nicht geändert zu werden braucht, und da es von Vorteil ist, wenn das Programm gleich nach Einschalten der Versorgungsspannung für den Einsatz im Universalzähler sofort verfügbar ist. Der ROM besteht in unserem Fall aus einer Diodenmatrix und aus einem dazugehörigen Decodierer. Man könnte für den ROM natürlich auch eine vollintegrierte Version verwenden, dann aber müßte man diesen von einer Firma (oder eventuell selbst) programmieren lassen. Diesen programmierbaren ROM nennt man auch PROM.

Auf das Programm, das der ROM enthält und welches die Funktionen des Universalzählers bestimmt, kommen wir später noch einmal zurück.

Die äußere starke Umrandung in der Schaltung beinhaltet die eigentliche Mikroprogrammsteuereinheit.

Der Zähler Z 5 übernimmt dabei die Funktion eines Programmzählers. Er hat Setzeingänge, so daß die Ausführung von Sprunganweisungen ermöglicht wird. ST 2 und ST 3 sind zwei 4-bit-Zwischenspeicher. Nachdem der Programmzähler eine Adresse an den ROM gibt, gelangt an die Zwischenspeicher der Befehlscode. Dieser Befehlscode muß nun in den Zwischenspeichern so lange festgehalten werden, bis dieser Befehl ausgeführt wurde. Durch dieses Vorgehen wird verhindert, daß der nun sofort neu ausgegebene nächste Befehlscode gleich wieder unkontrollierbar ausgeführt wird. Denn der Sprungbefehl ändert den Inhalt des Programmzählers und dann auch die Adresse für den ROM.

Die Ablaufsteuerung der Mikroprogrammsteuereinheit übernimmt ein nicht überlappender Zweiphasentakt. Der Taktimpuls Φ 1 bewirkt zunächst mit der ansteigenden Flanke ein Erhöhen des Inhalts des Programmzählers Z 5. Mit der abfallenden Flanke dieses Taktes wird der am Ausgang des ROMs stehende Befehlscode endgültig in die beiden Zwischenspeicher ST 2 und ST 3 übernommen. Mit dem Taktimpuls Φ 2, der nun anschließend auf der anderen Taktleitung erscheint, wird die Ausführung des Befehls bewirkt. Zur Entschlüsselung des Befehls dient ein Umcodierer M 2 des Typs SN 74154. Die Umcodierung wird durch den Taktimpuls Φ 2 freigegeben. Für die Ausführung von Sprungbefehlen zum Beispiel benötigt man noch Sprungadressen. Diese Sprungadresse wird durch den Inhalt des Zwischenspeichers ST 3 bestimmt. Die Adresse wird dazu an die Setzeingänge des Programmzählers geführt.

Um die weiteren Einzelheiten des Funktionsablaufs erklären zu können, muß nun die umliegende Schaltung in Betracht gezogen werden.

Beispielsweise gilt für die Ausführung eines unbedingten Sprungbefehls folgendes: Dieser Sprungbefehl liege codiert an den Ausgängen von ST 2. In der hier ausgeführten Schaltung ist die Zahl 0100_2 ($= 4_{10}$) der entsprechende Maschinencode für einen unbedingten Sprung. Als Adresse sei die Anweisung mit der Nummer 9 anzuspringen. Der vollständige Maschinencode lautet also dual 01001001 oder im Sedezimalsystem 49, oft auch Hexadezimalsystem genannt.

Wenn der Taktimpuls Φ 2 erscheint, gelangt ein Impuls über den Ausgang 4 des Umcodierers M 2 (auch Instruktionsdecoder genannt) an den Inverter I 9 und dann an die als Inverter geschaltete NAND-Verknüpfung N 16. Von dort aus gelangt der Impuls über eine „wired-or"-Verknüpfung an den mit „load" bezeichneten Eingang des Programmzählers. Der Impuls bewirkt dann, daß die Adresse, auf die ge-

sprungen werden soll (hier 9), in den Programmzähler geladen wird. Nun erscheint als nächstes wieder der Taktimpuls Φ 1, der ein Hochzählen des Programmzählers bewirkt. Die nächste Anweisung ist durch die um eins erhöhte Sprungadresse bestimmt und wird bei Erscheinen des Φ-2-Taktes ausgeführt. Der Rest der Schaltung kann am besten anhand der restlich vorhandenen Befehle erklärt werden:

Alles rücksetzen Code: Ex

Die Ausführung dieses Befehls bewirkt die Rücksetzung aller Flipflops in die Ruhestellung.

Der Ausgang E des Instruktionsdecoders ist deshalb mit den Rücksetzeingängen der RS-Flipflops verbunden. Über einen Inverter I 10 gelangt der Ausgang E des Instruktionsdecoders auch an die Rücksetzeingänge der Zähler.

In der Schaltung gibt es nämlich außer dem Programmzähler zwei weitere nicht direkt zusammenhängende Zählereinheiten. Der Zähler Z 4 hat die Aufgabe, die Adresse für den RAM-Speicher ST 1 zu bestimmen. Der RAM dient dabei als Hauptspeicher für die Anzeigeeinheit. Die Adresse von Z 4 bestimmt dann die Stelle im RAM, an die eine neue Dateninformation für die Anzeige eingeschrieben werden soll.

Ferner existiert noch die Zählerkette, bestehend aus Z 1, Z 2 und Z 3. Sie hat die Aufgabe, eine variable Verzögerungszeit zu bestimmen, und dient als Schleifenzähler.

Erhöhe den Adreßzähler Code: Dx

Mit diesem Befehl wird der Inhalt des Zählers Z 4 um eins erhöht. Um eine Rückmeldung vom Zähler Z 4 zu ermöglichen und so den Ablauf des Programms in Abhängigkeit vom Zähler Z 4 zu bringen, ist folgender Befehl vorgesehen:

Springe, falls kein Übertrag Code: Ax

Dabei wird der Sprungbefehl ausgeführt, wenn der Ausgang D des Zählers Z 4 auf LOW-Potential liegt. Dieses Ausgangssignal gelangt an den Inverter I 5, wird dort invertiert und gelangt dann an den einen Eingang der NAND-Verknüpfung N 14. Liegt der Ausgang D des Zählers also auf LOW-Potential, so gelangt an den einen Eingang der NAND-Verknüpfung N 14 ein HIGH-Signal. Wird nun ein Sprungbefehl Ax ausgeführt, so gelangt ein LOW-Impuls an den Eingang des Inverters I 6. Am anderen Eingang der NAND-Verknüpfung N 14 liegt ebenfalls ein HIGH-Signal. Damit ist die Verknüpfungsbedingung aber erfüllt und es gelangt ein LOW-Signal an den „load“-Eingang des Programmzählers. Damit wird der Sprungbefehl ausgeführt. Führt der Ausgang des Zählers Z 4 aber ein HIGH-Signal, so gelangt an den Eingang der NAND-Verknüpfung N 14 ein LOW-Signal. Ein LOW-Impuls am Eingang von I 6 bewirkt zwar einen HIGH-Impuls am anderen Eingang der NAND-Verknüpfung N 14, aber die Verknüpfungsbedingung ist diesmal nicht erfüllt und der Ausgang der NAND-Verknüpfung bleibt diesmal im alten Zustand (Offenen Kollektor). Ein Sprungbefehl wird dann nicht ausgeführt.

Für die Zählerkette Z 1, Z 2, Z 3 stehen ähnliche Befehle zur Verfügung:

Erhöhe Inhalt der Zählerkette und springe, falls kein Übertrag entstanden ist Code: 1x

Dieser Befehl ist ähnlich dem Befehl Ax. Nur daß zunächst einmal der Inhalt der Zählerkette um eins erhöht wird. Ein Sprung erfolgt nur, wenn der Ausgang D des Zählers Z 3 LOW-Potential führt. Dazu ist der Ausgang 1 des Instruktionsdecoders mit dem Eingang eines Inverters I 11 verbunden. Der Ausgang dieses Inverters ist mit dem Eingang des Zählers Z 1 verbunden. Wird der Befehl ausgeführt, so gelangt an den Eingang dieses Zählers ein HIGH-Impuls, dessen ansteigende Flanke ein Erhöhen des Inhalts der Zählerkette um eins bewirkt. Außerdem gelangt der Ausgang 1 des Instruktionsdecoders auch noch an den Eingang des Inverters I 8. Sein Ausgang führt

dann an den Eingang einer NAND-Verknüpfung N 15. Dabei bestimmt der Zustand des anderen Eingangs dieser NAND-Verknüpfung, ob ein Sprungbefehl ausgeführt wird oder nicht. Dazu wird diesem anderen Eingang über den Inverter I 7 die Information über den Zustand des Ausgangs D der Zählerkette zugeführt.

Für die Zählerkette stehen noch drei weitere Befehle zur Verfügung. Sie ermöglichen ein definiertes Setzen eines jeden der drei Zähler auf einen beliebigen Wert. Sie werden wie folgt ausgeführt.

Lade A **Code: 2x**

Die Ausführung dieses Befehls lädt den Zähler A (Z 1) mit dem im Adreßteil angegebenen Wert. Dazu werden die vier Datenleitungen, wie im Schaltbild eingezeichnet, mit den Setzeingängen der Zählerkette verbunden. Der Ausgang 2 des Instruktionsdecoders ist außerdem mit dem Ladeeingang des Zählers Z 1 verbunden. Ein LOW-Impuls, wie er bei der Ausführung des Befehls entsteht, lädt den Zähler Z 1 mit dem auf den Datenleitungen vorhandenen Wert.

Lade B **Code: 3x**

Die Ausführung dieses Befehls lädt den Zähler B (Z 2) mit dem im Adreßteil gegebenen Wert.

Lade C **Code: 5x**

Hier wird bei der Ausführung der Zähler C (Z 3) mit dem im Adreßteil gegebenen Wert besetzt.

Damit wären alle Befehle, die die Zähler beeinflussen, genannt.

Die nächste Gruppe von Befehlen befaßt sich mit dem RAM-Speicher. Dieser hat eine Kapazität von 64 x 4 bit und ist in dem Schaltbild mit ST 1 bezeichnet.

Schreibe in den Speicher und nach Peripherie **Code: Fx**

Hiermit ist es möglich, ein am Eingang des Speichers stehendes 4-bit-Datenwort in den Speicher an die Stelle zu schreiben, die durch die 4-bit-Adresse am Adreßeingang des Speichers bestimmt ist. Die Adreßeingänge des Speichers sind dabei mit den Ausgängen eines Multiplexers verbunden. Dieser hat die Möglichkeit, entweder die Adresse von den Eingängen BA, BB, BC und BD auf seinen Ausgang zu schalten, oder die Adresse, die an den Eingängen AA, AB, AC und AD steht. Die Eingänge BA, BB, BC und BD sind mit den Ausgängen des Zählers im Anzeigeteil verbunden und die Eingänge AA, AB, AC und AD mit den Ausgängen des Zählers Z 4. Die Umschaltung des Multiplexers wird über den „select"-Eingang des Multiplexers bestimmt. Dabei liefert der Zustand des RS-Flipflops, bestehend aus N 7 und N 8, die jeweilige Information. Wird der Rücksetzbefehl Ex ausgeführt, so gelangt ein LOW-Impuls an die NAND-Verknüpfung N 7 und der Ausgang dieses RS-Flipflops geht auf HIGH. Dadurch wird erreicht, daß die B-Eingänge des Multiplexers auf seine Ausgänge durchgeschaltet werden. Um ein Umschalten auf die Eingänge A zu ermöglichen, steht der Befehl zur Verfügung:

Setze MUX auf internes System **Code: Cx**

Die Eingänge AA, AB, AC sind dabei mit dem Zähler Z 4 verbunden. Der Eingang AD liegt an einem Ausgang des RS-Flipflops mit N 9 und N 10. Es ist somit möglich, die Daten mit den Adressen 0...7 getrennt von den Daten mit den Adressen 8...15 einzuspeichern. Nach Ausführung des Befehls mit dem Code Ex steht das Flipflop N 9, N 10 so, daß die Daten mit den Adressen 0...7 angesprochen werden.

Um das Flipflop in den anderen Zustand kippen zu lassen, gibt es den Befehl:

Datenadreßumschaltung **Code: Bx**

Damit wird dieses Flipflop so gesetzt, daß die Adressen 8...15 des RAMs ST 1 angesprochen

werden. Es steht dem Benutzer noch das RS-Flipflop N 5 und N 6 zur Verfügung. Dieses Flipflop ermöglicht eine Freigabe der Datenausgänge, die für externe Geräte zur Verfügung stehen.

Ausgangsverknüpfungen öffnen Code: 9x

Mit diesem Befehl werden die NAND-Verknüpfungen N 1...N 4 freigegeben.

Lösche Frequenzzähler Code: 6x

Dieser Befehl setzt die Zähleinheit, die im Abschnitt 3.2 beschrieben wurde, auf Null.

Gib Zählereingang frei Code: 7x

Damit ist es möglich, das in der Schaltung vorhandene NAND-Schaltglied N 13 mit Hilfe des RS-Flipflops N 11 und N 12 freizugeben, dessen Ausgang an den Zähleingang bei entsprechender Stellung des Schalters S 1 führt. An den Eingang N 13 kann man zum Beispiel die zu messende Frequenz legen. Mit dem Schalter S 1 kann man einstellen, ob der Zähleingang und der Rücksetzeingang der Zählereinheit extern zugänglich sein soll oder nicht. Mit S 2 kann man noch bestimmen, ob der Rücksetzvorgang trotzdem automatisch erfolgen soll.

Um den Universalzähler zu betreiben, benötigt man nun noch das Steuerungsprogramm.

Der Leser kann später anhand des angegebenen Programms, mit dem er Erfahrung sammeln kann, und den zusätzlichen Befehlen, die ihm zur Verfügung stehen, ein eigenes Programm gestalten. Dieses Programm kann nur eine Erweiterung vorstellen oder nach eigenen Wünschen dem Universalzähler völlig andere Eigenschaften verleihen. Die Aufgabenstellung für das Programm, wie es hier gegeben wird, lautet nun aber folgendermaßen.

Ist S 1 betätigt, soll man den Universalzähler beliebig verwenden können. Es stehen dann drei Eingänge zur Verfügung, wenn S 2 nicht betätigt ist, andernfalls nur die ersten beiden:

(1) Latch-Eingang:
Ein LOW-Impuls von bestimmter Dauer an diesem Eingang gibt den Befehl, die Daten der Zählereinheit in der Anzeigeeinheit anzuzeigen.

(2) Zähl-Eingang:
Hiermit kann der Inhalt der Zählereinheit um die Anzahl an eingegebenen Impulsen erhöht werden.

(3) Rücksetz-Eingang:
Damit ist es möglich, die Zählereinheit auf Null zu setzen.

Das Programm für den Universalzähler hat nun die Aufgabe, zunächst die Daten aus der Zählereinheit in den Speicher ST 1 zu schaffen, damit der Zählerstand angezeigt werden kann. Dann muß die Zählereinheit, falls S 1 nicht betätigt ist (oder S 1 betätigt ist und S 2 nicht), auf Null gesetzt werden. Anschließend wird N 13 geöffnet, so daß eine Frequenzzählung durchgeführt werden kann. Als nächstes wird eine Warteschleife durchlaufen, die eine definierte Frequenzmessung ermöglicht.

Ist die Verzögerungszeit beendet, die durch diese Warteschleife erzeugt wurde, so springt die Ausführung des Programms wieder an den Anfang zurück. Dort wird der Rücksetzbefehl gegeben und somit auch N 13 wieder gesperrt. Anschließend erfolgt wieder die Übertragung des Zählerinhalts auf den Speicher ST 1. *Abb. 3.4-4* zeigt das dazugehörige Flußdiagramm. Das entsprechende Maschinenprogramm sieht dann folgendermaßen aus:

Adresse:	Code:	Erklärung:
0	Ex	Alles rücksetzen
1	9x	Ausgänge freigeben für externe Ausgabe der Daten
2	Cx	Multiplexer M 1 auf internen Zähler Z 4 umschalten

3	Fx	In den Speicher die Daten einschreiben
4	Dx	Adreßzähler Z 4 um eins erhöhen
5	A2	Falls kein Übertrag zurück zu Adresse 2 (Adresse 3 wird als nächstes ausgeführt)
6	Ex	Alles rücksetzen
7	6x	Zählereinheit auf 0 setzen
8	2F	Zähler A mit F laden
9	37	Zähler B mit 7 laden
A	51	Zähler C mit 1 laden
B	7x	N 13 freigeben
C	00	Verzögerung (Nulloperation)
D	1A	Falls kein Übertrag zurück nach Adresse A und außerdem zuvor Zählerketteninhalt um eins erhöhen. (Falls Sprung erfolgt, wird die Ausführung erst bei Adresse B vorgenommen)
E	00	Verzögerung
F	00	Verzögerung

Ein Rücksprungbefehl am Ende des Programms erübrigt sich, da die Zählkapazität des Programmzählers auf 4 bit beschränkt ist und somit nach Ausführung des Befehls mit der Adresse 1111_2 (= F_{16}) wieder die Adresse 0000_2 (= 0_{16}) angewählt wird.

Im Schaltbild 3.4-3 ist das Programm in den ROM schon eingezeichnet. Dabei werden die Stellen mit einer Diode entsprechend dem Schaltbild besetzt, die im dualen Code des Maschinenprogramms eine 1 haben.

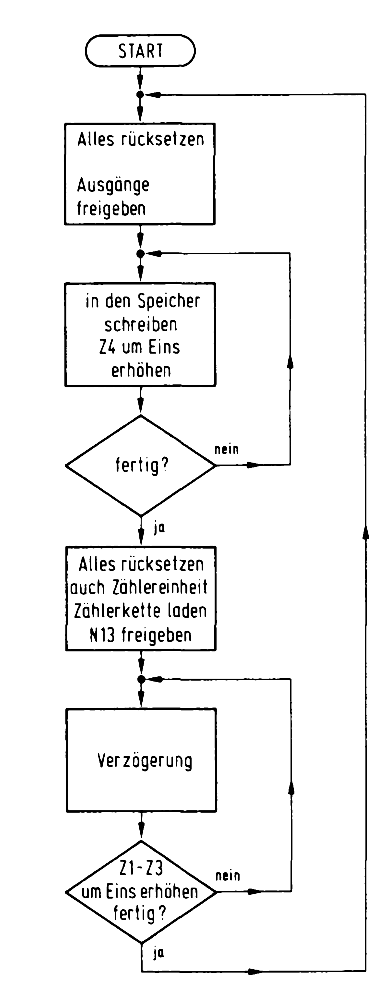


Abb. 3.4-4 Flußdiagramm des Steuerprogramms

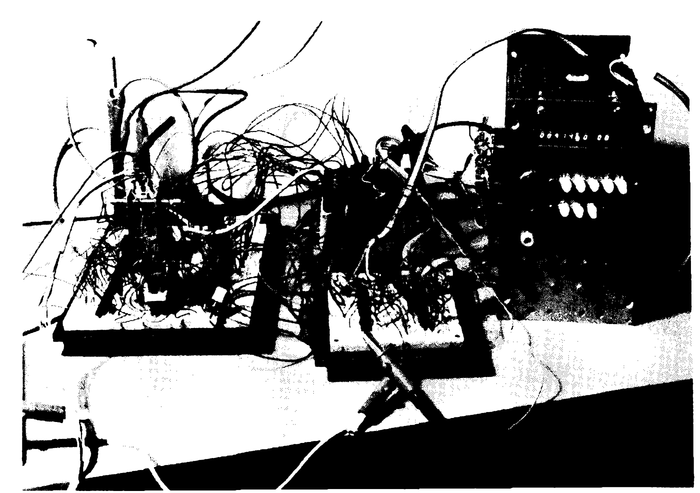


Bild A: Versuchsaufbau des Mikroprogrammgesteuerten Universalzählers. mit dem im Kapital 3.4 beschriebenen Programm für den Einsatz des Universalzählers als Frequenzmeßgerät



Bild B: Möglicher Aufbau des Datensichtgerätes.

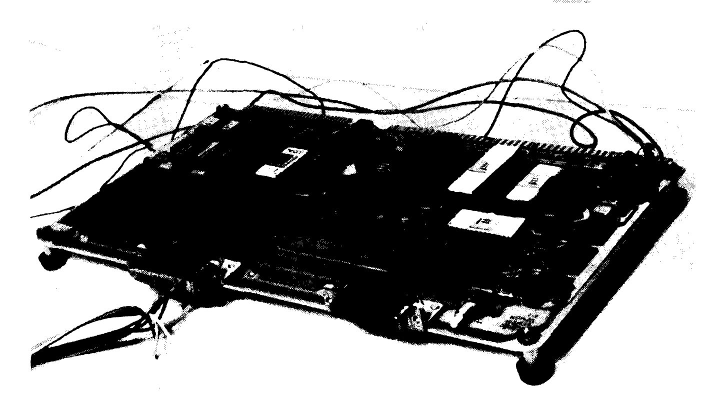


Bild C: Mikrocomputer mit dem 8080. (SDK 80 Kit von Intel)

4 Ausgabesysteme

In diesem Kapitel sollen zunächst einmal Ausgabesysteme besprochen werden, die in der Lage sind, alphanumerische Zeichen auf einem Bildschirm darzustellen.

Im Abschnitt 4.1 wird der Bildschirm eines Oszilloskops verwendet, im Abschnitt 4.2 der Bildschirm eines Fernsehgerätes.

4.1 Anzeige von alphanumerischen Zeichen auf einem Oszilloskop

Hier soll das Aufbauprinzip einer Schaltung erklärt werden, die es ermöglicht, in 16 Spalten mit 8 Zeilen insgesamt 128 Zeichen auf einem Oszilloskop darzustellen.

Das Oszilloskop benötigt lediglich einen x- und einen y-Eingang (aber keinen z-Eingang).

4.1.1 Punktrastergenerator

Abb. 4.1-1 zeigt das gesamte Prinzipschaltbild eines solchen Gerätes.

Im Schaltbild ist der Punktrastergenerator durch den stark umrandeten Teil hervorgehoben. Er hat die Aufgabe, den Bildschirm mit einem 96 x 64 Punkte fassenden Raster zu überziehen. Die Position eines jeden Punktes wird durch die Spannungen, die an die x- und y-Eingänge des Oszilloskops geführt werden, bestimmt. Die eigentliche Festlegung der Position eines Bildpunktes geschieht durch die Zähler Z 1...Z 4.

Dabei bestimmen die Zähler Z 1 und Z 2 die Position in x-Richtung. Die Zähler Z 3 und Z 4 bestimmen die Position in y-Richtung.

Ein Zeichen besteht aus 6 x 8 Punkten einschließlich des Zwischenraums für das nächste daneben und darunter liegende Zeichen. Der Zähler Z 1 bestimmt in x-Richtung die Auswahl eines der 6 Punkte, die zur Darstellung eines Zeichens notwendig sind. Es ist daher erforderlich, diesen Zähler so aufzubauen, daß dieser von 0...5 zählt. Die Ausgabe der Positionsinformation geschieht im Dual-Code. Diese Information wandert dann an einen D/A-Umsetzer, der ein der Position proportionales Spannungssignal gibt. Der Zähler Z 1 muß ferner ein Übertragsignal liefern, wenn ein Sprung von 5 auf 0 erfolgt. Dieses Übertragsignal steuert dann den Zähler Z 2, indem er ihn veranlaßt, den Zählerinhalt um eins zu erhöhen. Dieser Zähler zählt dann im Dual-Code von 0...15. Dieser Zähler Z 2 bestimmt in x-Richtung die Position des Zeichens, das gerade angezeigt werden soll. Die Information über den Zählerstand gelangt ebenfalls an einen D/A-Umsetzer und wird dort in eine der Position proportionale Spannung umgewandelt.

Diese Spannung des D/A-Umsetzers mit der Bezeichnung D 2 wird mit der Spannung, die der D/A-Umsetzer D 3 liefert, über ein Potentiometer P gemischt. Es wäre nicht sinnvoll, einen einzigen D/A-Umsetzers für die Umwandlung der Zählerinformationen Z 1 und Z 2 zu verwenden. Der Zähler Z 1 soll ja

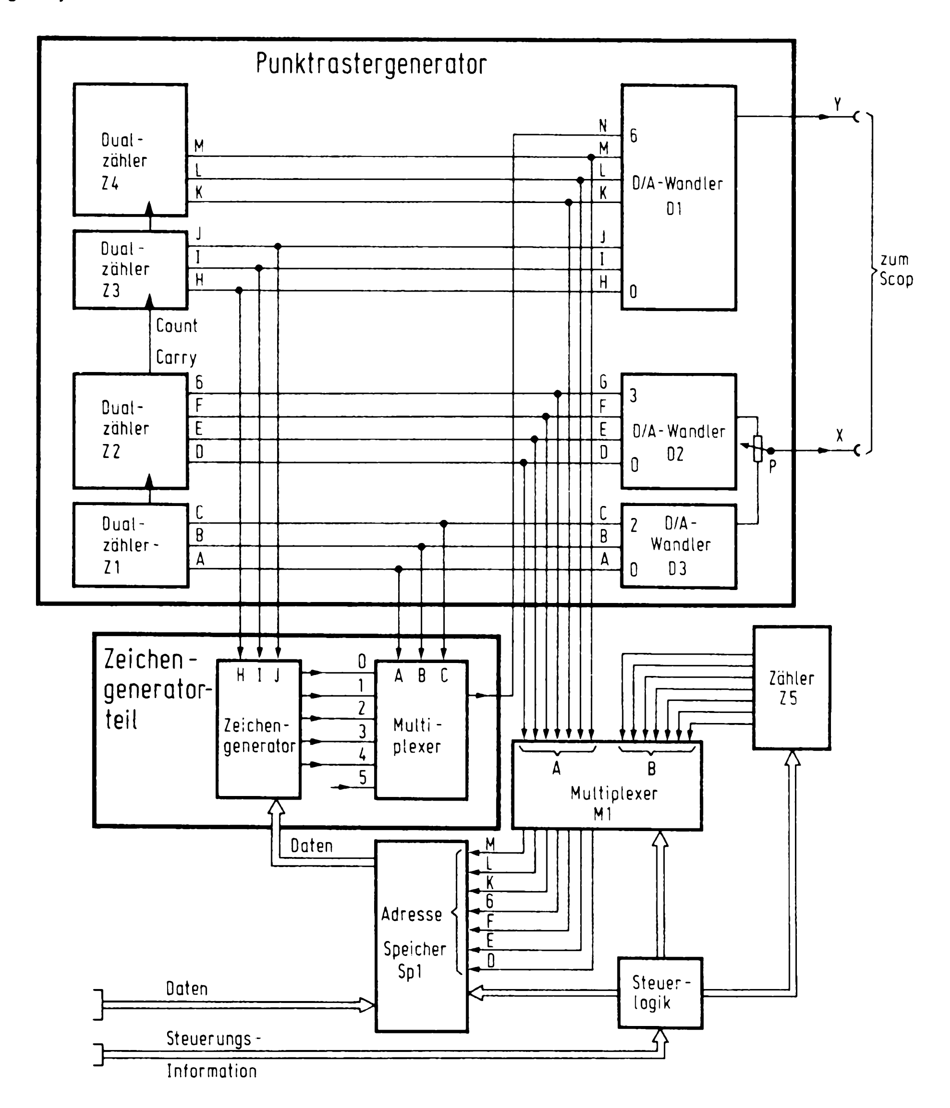


Abb. 4.1-1 Prinzipschaltbild des Anzeigegerätes

nur bis 5 zählen und nicht, wie es die volle Ausnützung der drei Bits ermöglichen würde, bis 7. Der D/A-Umsetzer würde aber eine Umwandlung gemäß dem Dual-Code vornehmen.

Dadurch würde der Abstand zwischen zwei Zeichen in x-Richtung aber unnötig groß werden.

Dual gesehen besteht nämlich zwischen zwei Zeichen in x-Richtung ein Sprung. Zum Beispiel zwischen dem Zeichen mit der Adresse 0 und dem Zeichen mit der Adresse 1, in x-Richtung.

Der Code für den letzten Punkt des Zeichens mit der Adresse 0 lautet:

0000101

und der Code für den nächsten Punkt, also den ersten Punkt des nächsten Zeichens lautet:

0001000.

Durch die getrennte Umwandlung der Information von Z 1 und Z 2 und die anschließende Mischung mit dem Potentiometer P ist es aber möglich, durch die entsprechende Stellung des Potentiometers zu erreichen, daß ein lückenloser Übergang zwischen den Zeichen in x-Richtung besteht. Der kleine Zwischenraum zwischen den eigentlichen Zeichen ist dabei natürlich noch vorhanden. Hier wurde als Zeichen die ganze Punktreihe von 6 Punkten in x-Richtung angesehen.

Der Zähler Z 3 bestimmt die Position des Punktes eines Zeichens in y-Richtung. Dazu muß er von 0...7 zählen. Die Information gelangt dann dual an den D/A-Umsetzer D 1. Mit dem Zähler Z 4 wird schließlich noch die Position des Zeichens in y-Richtung bestimmt. Dazu muß dieser Zähler ebenfalls von 0...7 zählen können. Die Information über den Zählerstand von Z 4 gelangt auch an den D/A-Umsetzer D 1. Diesmal ist es möglich, für beide Zähler einen D/A-Umsetzer zu verwenden, da sich keine Sprünge ergeben und der Dual-Code voll ausgeschöpft wird.

Der D/A-Umsetzer D 1 besitzt noch einen weiteren Eingang, der mit N bezeichnet ist.

Dieser Eingang ist mit dem Ausgang des Zeichengeneratorteils verbunden. Der logische Zustand dieses Signals liefert die Information, ob ein Bildpunkt erscheinen soll oder nicht.

Zur Erklärung wird ein Bildpunkt mit der Adresse 0000100 in x-Richtung und mit der Adresse 000011 in y-Richtung betrachtet. Soll der Punkt erscheinen, so liegt am Eingang N des D/A-Umsetzers D 1 das Signal 0. Der vollständige Code am D/A-Umsetzer D 1 lautet also dann 00000100. Wenn nun aber ein Bildpunkt nicht erscheinen soll, so liegt am Eingang N das Signal 1. Dann lautet der vollständige Code am D/A-Umsetzer D 1 1000011. Da am Ausgang des D/A-Umsetzers ein dem Betrag des Codes proportionales Analog-Signal entsteht, und der Code des Punktes, der dargestellt werden soll, sich von dem Punkt, der nicht dargestellt werden soll, in seiner höherwertigen Stelle unterscheidet, unterscheidet sich auch das Analogsignal der beiden Bildpunkte durch eine dem höherwertigen Bit entsprechenden Spannung. Wenn also ein Bildpunkt nicht erscheinen soll, so wird mit diesem Kunstgriff zu dem y-Signal eine Spannung addiert, die den Punkt bei entsprechend eingestelltem Ablenkkoeffizienten des Oszilloskops in y-Richtung in einen nicht dargestellten Bereich befördert. Damit ist es möglich, auf einen z-Eingang am Oszilloskop zu verzichten. *Abb. 4.1-2* zeigt das Bildschirmraster für den Fall, daß alle Bildpunkte dargestellt werden. Die dunkel dargestellten Punkte bezeichnen dabei die Stellen, an denen später die Zeichenzwischenräume durch „Dunkeltasten", also durch ein entsprechendes 1-Signal am Eingang N (Abb. 4.1-1) erzeugt werden.

4.1.2 Zeichengeneratorteil

Dieser in Abb. 4.1-1 unten stark umrandete Teil bestimmt das Aussehen der darzustellenden Zeichen. Die Industrie liefert eine Vielfalt solcher Zeichengeneratoren, angefangen von 64 möglichen Zeichen, die darstellbar sind im 5 x 7 Raster, bis zu Zeichengeneratoren mit 128 darstellbaren Zeichen im 7 x 9 Raster.

Abb. 4.1.2-1a...d zeigt die darstellbaren Zeichen bei dem von Texas Instruments lieferbaren Zeichengenerator TMS 2501NC.

Der Zeichengenerator erhält die Information von dem Speicherteil darüber, welches Zeichen gerade dargestellt werden soll. Die Information, welche Zeile des Zeichens ausgegeben werden soll, liefert der Zähler Z 3 an den Zeichengenerator. Als Ausgangssignale gibt der Zeichengenerator parallel die 5 Punkte der angewählten Zeile und des angewählten Zeichens aus. Diese Daten gelangen an einen Multiplexer, der die Aufgabe hat,

Abb. 4.1-2 Dargestelltes Punktraster

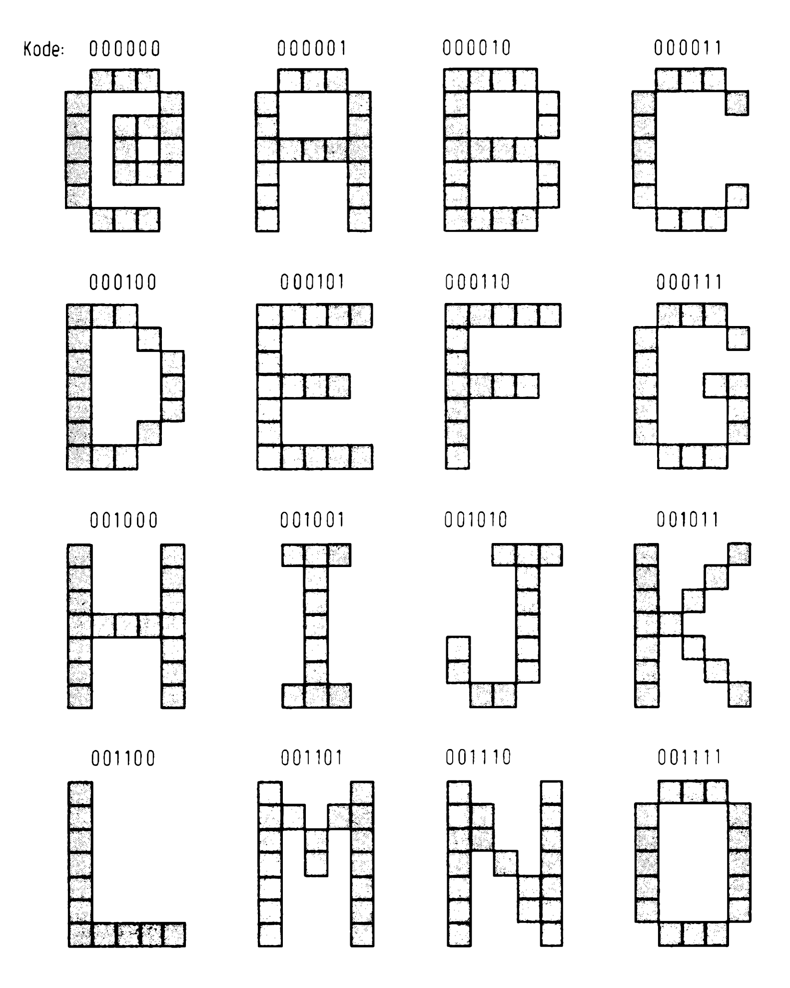


Abb. 4.1.2-1a–d Vom Zeichengenerator TMS 2501 darstellbare Zeichen

das gewünschte Dunkeltastsignal zu liefern. Dazu erhält er die Information, welche Spalte des Zeichens angewählt ist, über die Leitungen des Zählers Z 1. Der Zähler Z 1 kann 6 verschiedene Ausgangskombinationen haben. Zu jedem der 6 Möglichkeiten wählt der Multiplexer, dem Dual-Code folgend, einen seiner 6 Eingänge aus und schaltet diesen an seinen Ausgang. Hier kann man auch verstehen, wie der Zwischenraum in x-Richtung erzeugt wird. Er entsteht durch den offen gelassenen 6. Eingang des Multiplexers. In y-Richtung entsteht auch ein Zwischenraum, doch wird dieser intern im Zeichengenerator erzeugt.

4.1.3 Speicher mit Schieberegister

Die Speicherung der Daten mit einem Schieberegister bringt einige Vorteile. Es ist zum Beispiel keine spezielle Steuerschaltung notwendig, um ein sequentielles Einspeichern der Daten zu ermöglichen. Aber die Speicherung der Daten mit einem Schieberegister hat auch Nachteile. So ist es zum Beispiel notwendig,

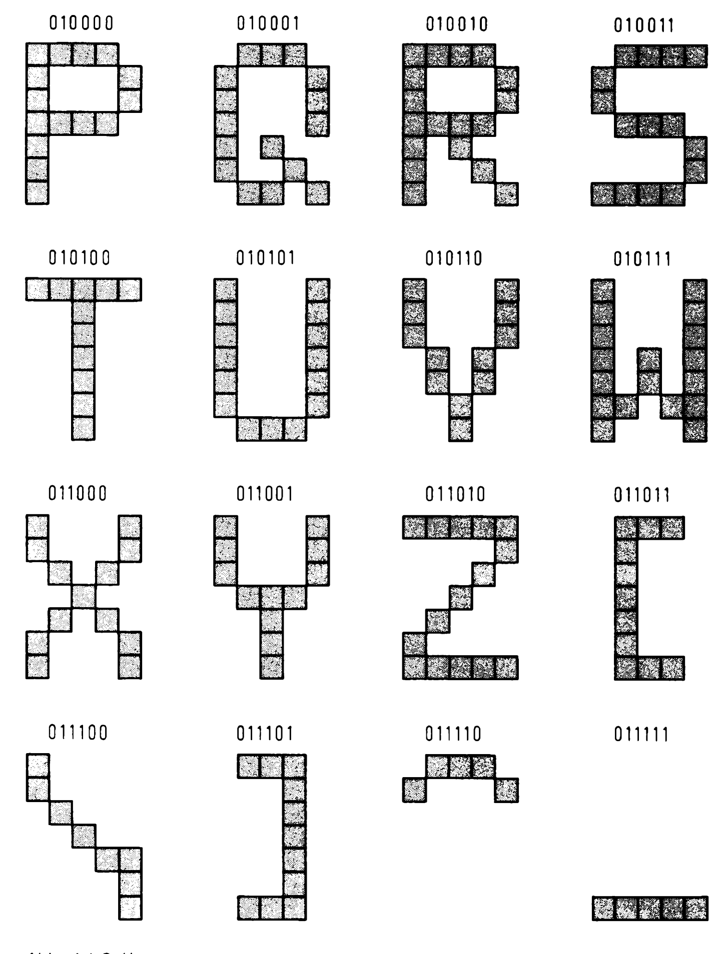


Abb. 4.1.2-1b

die Daten parallel mit einem Multiplexer abzutasten oder ein Umschalten von Abtastvorgang auf Einspeichervorgang zu steuern. Beides bringt einen hohen Hardwareaufwand mit sich. Ferner lassen sich Sonderbefehle wie „carriage return", „line feed", „up", „backstep", „down" etc. nur schwer realisieren (siehe Tabelle 4.2.9-1).

4.1.4 Speicher mit RAM

Diese Speichermethode stellt wohl die eleganteste dar. Heute sind derartige Speicher mit sehr hohen Speicherkapazitäten zu sehr günstigen Preisen im Handel. Verwendet wird am besten ein statischer Speicher mit einer n x 4- oder besser mit einer n x 8-Organisation, das heißt ein Speicher, bei dem 4 bzw. 8 bit für eine eingegebene Adresse parallel ausgegeben werden.

4.1.5 Steuerteil für RAM

Der Steuerteil muß die ankommenden Daten an die richtige Position im Speicher schreiben und spezielle Befehle („carriage return", „line feed"...) erkennen und ausführen.

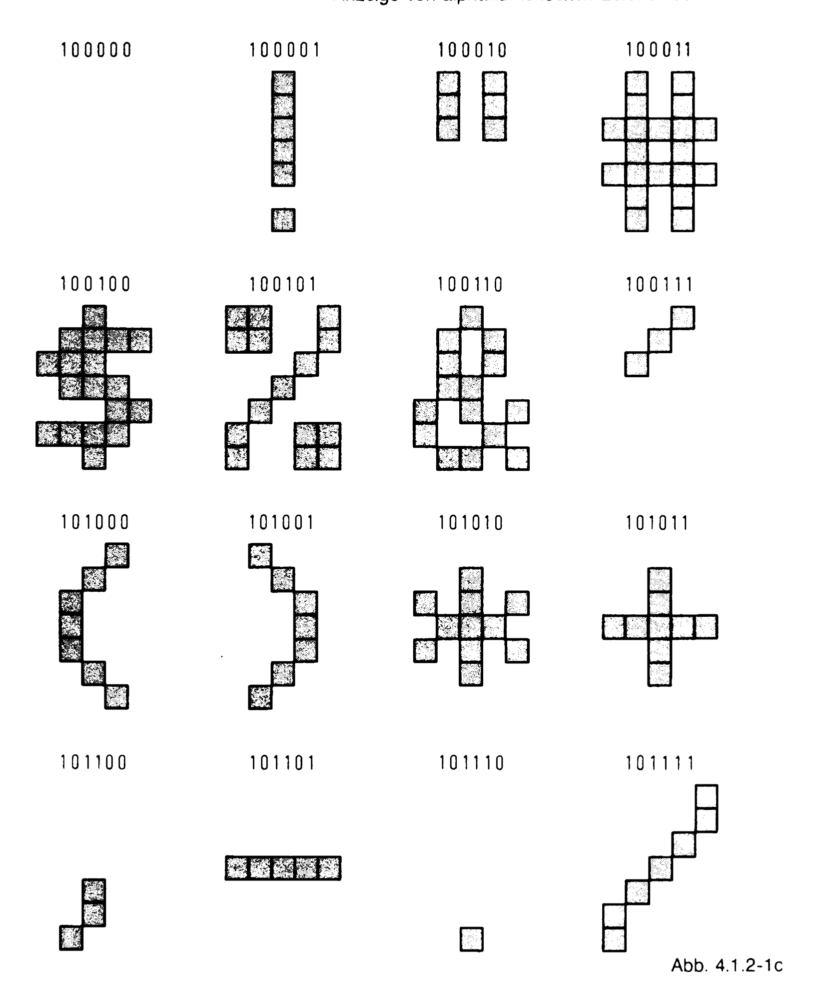


Abb. 4.1.2-1c

Um die Position bestimmen zu können, in der im Speicher das nächste Zeichen eingespeichert werden soll, muß zunächst einmal die Adresse der Position gespeichert werden. Dies geschieht am einfachsten mit einem Zähler (Z 5). Um im Normalbetriebszustand, in dem keine Anforderung auftritt, ein Zeichen einzuschreiben, muß der Speicherinhalt (Sp 1) permanent auf dem Oszilloskop dargestellt werden. Der Punktrastergenerator bestimmt ebenfalls eine Adresse, die in diesem Fall die jeweilige Position eines auszugebenden Zeichens repräsentiert. Diese Adressen müssen im Falle der permanenten Darstellung laufend an den Speicher gegeben werden. Wird nun ein Zeichen in den Speicher SP 1 eingegeben, so muß die vom Zähler Z 5 stammende Adresse an den Speicher gelangen. Um diese Umschaltung zu verwirklichen, ist ein Multiplexer M 1 vorgesehen. Die Steuerschaltung hat dann zu bestimmen, wann von den Adressen des Punktrastergenerators auf die Adressen des Zählers Z 5 umgeschaltet werden soll. Um die schon genannten Sonderbefehle ausführen zu können, ist es vorteilhaft, wenn der verwendete Zähler Z 5 die Möglichkeit besitzt, Adressen zu laden.

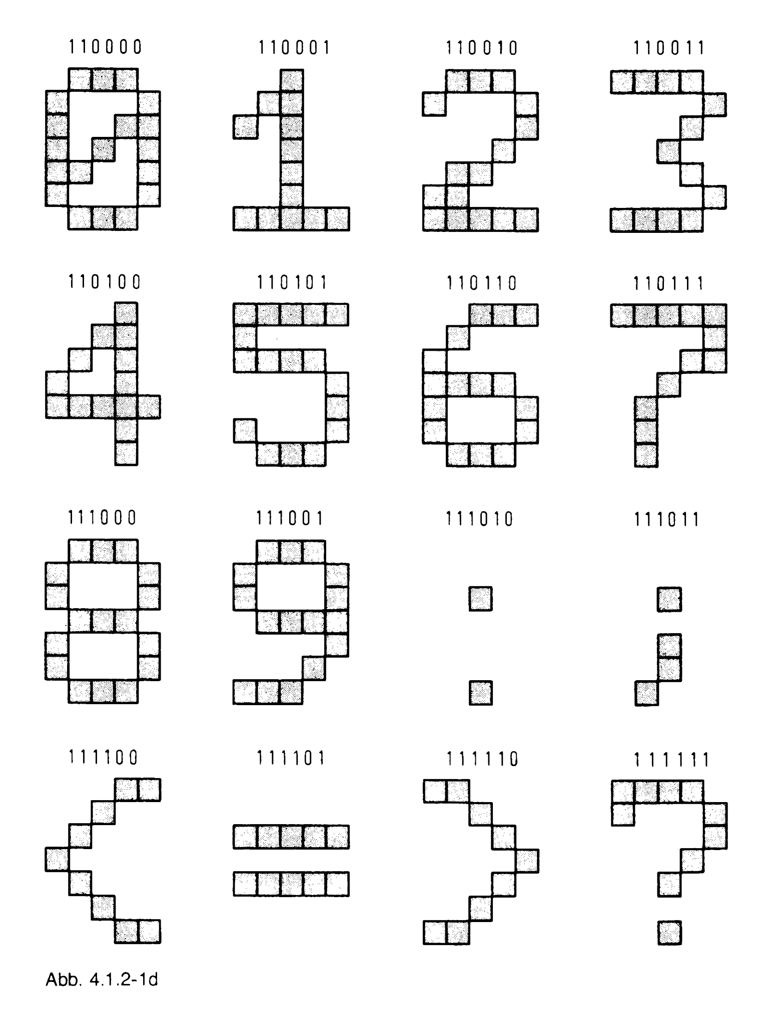


Abb. 4.1.2-1d

4.2 Datensichtgerät, Anzeige von alphanumerischen Zeichen auf dem Bildschirm eines Fernsehgerätes

Das hier beschriebene Datensichtgerät arbeitet über den Hf-Eingang eines gewöhnlichen Fernsehgerätes. Es gestattet die Darstellung von 64 Zeichen in einer Zeile und von 16 (in der Ausbaustufe 32) Zeilen. Die Daten können über ein standardmäßiges Signal, wie es zum Beispiel von „Mikrocomputererprobungskits" geliefert wird, eingegeben werden. Ebenfalls ist es möglich, die Daten parallel (im ISO-7-bit-Code = DIN 66 003) einzugeben. Da das Datensichtgerät auch eine Tastatur besitzt, wurde ein serieller Datenausgang vorgesehen und auch ein Parallelausgang. Genauer gesagt, steht ein Datenbus für die Ein- und Ausgabe der Information im Parallelformat zur Verfügung.

Die Schaltung des Datensichtgerätes sieht einen mit maximal 2 KByte belegten Speicher vor. Es ergibt sich aber bei dieser Speicherkapazität ein mit Zeichen dicht belegtes Bildfeld. Deshalb wurde zugunsten der Lesbarkeit die

Möglichkeit gegeben, diesen Speicher nur mit 1 KByte zu belegen und dabei die Zeichen so verteilt darzustellen, daß nach jeder beschreibbaren Zeile auf dem Bildschirm eine Zeile folgt, die nicht mit Zeichen vollgeschrieben werden kann. Dabei ergeben sich 16 gleichmäßig verteilte, beschreibbare Zeilen und somit eine gute Lesbarkeit der dargestellten Information.

Abb. 4.2-1 zeigt das Blockschaltbild des Datensichtgerätes. Entsprechend diesem Blockschaltbild werden die einzelnen Einheiten im Detail besprochen.

4.2.1 Hochfrequenzgenerator

Die in *Abb. 4.2.1-1* dargestellte Schaltung hat die Aufgabe, das sogenannte BAS-Signal (das später noch genauer erklärt wird) einem Hf-Träger aufzumodulieren, so daß es möglich wird, das Signal direkt einem gewöhnlichen TV-Gerät zuzuführen.

Die im Schaltplan eingezeichnete Spule L besteht aus vier Windungen 1 mm starken Kupferdrahts. Die Spule besitzt dabei einen Durchmesser von 8 mm. Mit dem Trimmer C5 ist es möglich, die Frequenz des Hf-Generators auf einen Kanal im VHF-Bereich einzustellen. Besitzt der Fernseher einen 60-Ω-Eingang, so kann der eingezeichnete Hf-Trafo entfallen.

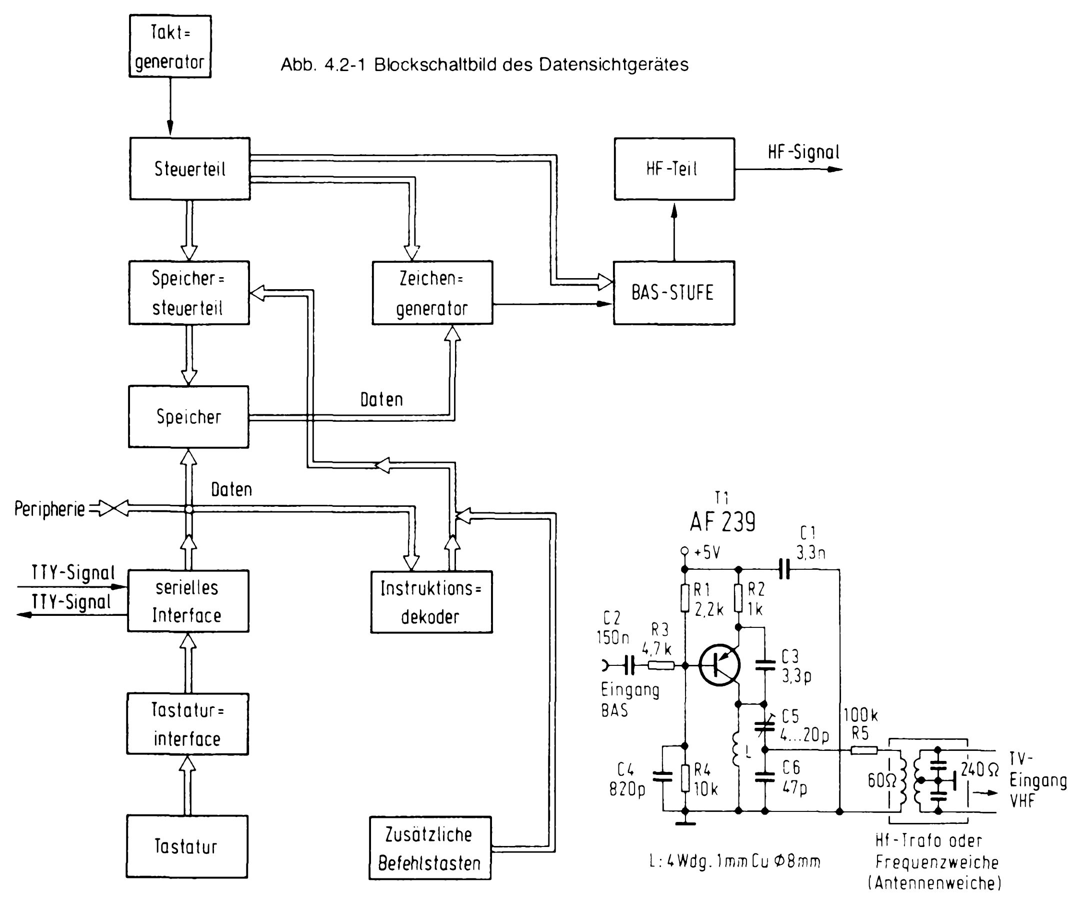


Abb. 4.2-1 Blockschaltbild des Datensichtgerätes

Abb. 4.2.1-1 HF-Generator

Ist ein 240-Ω-Eingang vorhanden, so ist die Verwendung eines Hf-Trafos unbedingt notwendig. Anstatt des Hf-Trafos kann man auch eine Frequenzweiche mit der Eingangsimpedanz von 60 Ω und mit der Ausgangsimpedanz von 240 Ω verwenden. An das Fernsehgerät ist dann der VHF-Ausgang der Frequenzweiche an den VHF-Eingang des Fernsehers anzuschließen. Der UHF-Ausgang der Frequenzweiche bleibt dabei unbeschaltet. Zum Test dieses Teils stellt man den Fernseher mit dem Kanalschalter grob auf die Frequenz des Hf-Generators ein und gleicht dann mit dem Trimmer C 5 die Frequenz so ab, daß der Bildschirm des Fernsehers dunkel erscheint. Der Eingang BAS des Hf-Generators braucht dazu noch kein Signal zu erhalten, der Eingang bleibt also unbeschaltet.

Damit ist der Abgleich der Hf-Stufe beendet.

4.2.2 BAS-Mischer

Abb. 4.2.2-1 zeigt die Schaltung des BAS-Mischers. Er hat die Aufgabe, das in *Abb. 4.2.2-2* dargestellte BAS-Signal aus drei ankommenden Komponenten zusammenzustellen.

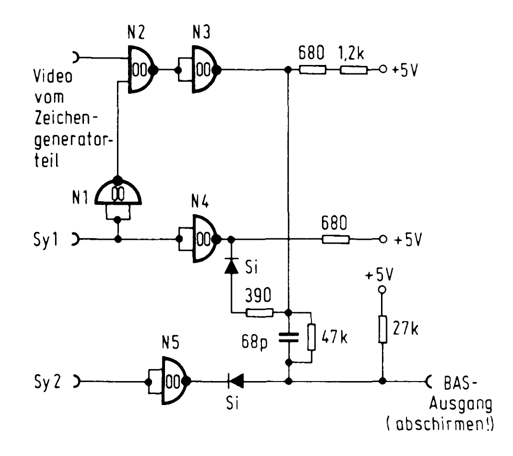


Abb. 4.2.2-1 BAS-Stufe

Das BAS-Signal kann dann direkt der Hf-Stufe zugeführt werden. Dieses BAS-Signal beinhaltet alle Informationen, die das Fernsehgerät braucht, um richtig synchronisiert zu werden, und es beinhaltet die eigentliche Bildinformation. Das Fernsehbild wird bei einer normalen TV-Sendung mit Hilfe des sogenannten Zeilensprungverfahrens zusammengesetzt. Dies bedeutet, zunächst wird beginnend bei der ersten Zeile des Fernsehbildes jede weitere Zeile geschrieben. Nachdem das erste Teilbild auf diese Weise geschrieben wurde, wird der Rest, also alle gradzahligen Zeilen des Fernsehbildes ebenfalls geschrieben.

Dieser Kunstgriff dient dazu, eine Rasterfrequenz (auch Bildfrequenz genannt) von 50 Hz zu erreichen und somit einen flimmerfreien Eindruck des dargestellten Bildes zu ermöglichen, während die eigentliche Bildwechselfrequenz nur 25 Hz beträgt. Der Steuerteil des Datensichtgerätes bewirkt allerdings ein BAS-Signal, das dieses Zeilensprungverfahren im TV-Gerät nicht erzwingt. Es wird also auch beim zweiten Teilbild, nur beginnend bei der ersten Zeile, angefangen, jede zweite Zeile zu schreiben, und somit wird immer nur jede zweite Zeile beschrieben. Damit decken sich hier beide Teilbilder.

Dies beeinträchtigt die einwandfreie Funktion des Gerätes aber nicht, sondern im Gegenteil bewirkt es, daß in vertikaler Richtung immer ein kleiner Zwischenraum zwischen den

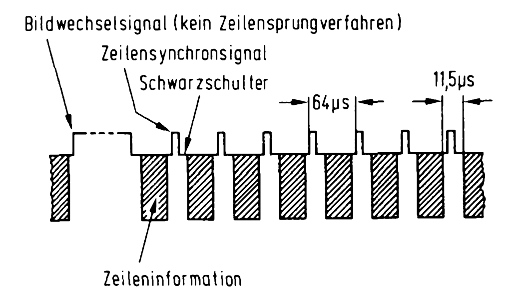


Abb. 4.2.2-2 BAS-Signal

einzelnen Bildpunkten bleibt. Die Erzeugung eines Synchronsignals, das das Zeilensprungverfahren erlaubt, hätte auch einen größeren Schaltungsaufwand bedeutet. In einer Schaltung, die später noch beschrieben wird, wird erreicht, daß auch in horizontaler Richtung jeweils ein Zwischenraum zwischen den einzelnen Bildpunkten bleibt. Insgesamt ist dadurch eine gute Lesbarkeit der angezeigten Zeichen gegeben.

Doch nun zurück zu der Beschreibung des BAS-Mischers. Ihm werden zwei Synchronsignale zugeführt, das Zeilensynchronsignal mit dem Bildwechselsynchronsignal, als Sy 2 bezeichnet, und ein mit Sy 1 bezeichnetes Signal, das die Schwarzschulter bewirkt und ferner dafür sorgt, daß ein Dunkelfeld den eigentlichen „Informationsraum" umgibt, in dem Zeichen dargestellt werden können. Das Signal Sy 1 wird zusätzlich über eine Nicht-Verknüpfung N 1 einer NAND-Verknüpfung N 2 zugeführt. Dieser NAND-Verknüpfung N 2 wird außerdem das Video-Signal zugeführt, das die Bildinformation trägt. Durch die NAND-Verknüpfung wird erreicht, daß eventuelle, nicht definierte Signalzustände des Video-Signals keine Fehlinformation verursachen. Nicht definierte Signalzustände können genau in der Zeit auftreten, in der sich der Strahl des TV-Gerätes an Stellen befindet, an denen kein Zeichen dargestellt werden kann.

Der Rest der Schaltung des BAS-Mischers dient dazu, die Synchronsignale und das Videosignal mit entsprechenden Pegeln zu einem BAS-Signal zu mischen.

4.2.3 Steuerteil, Erzeugung der Synchron-, Speicher- und Steuersignale

Der Steuerteil stellt gewissermaßen den Kern des Datensichtgerätes dar. Er hat die Aufgabe, alle wichtigen Signale zu erzeugen [3].

Der Takt, der die ganze Schaltung versorgt, wird zugunsten einer höheren Stabilität mit einem einfachen Quarzgenerator erzeugt. Der Generator schwingt dabei auf einer Frequenz von 8 MHz. Der Generator ist ebenfalls in *Abb. 4.2.3-1* eingezeichnet, das die Schaltung des ganzen Steuerteils zeigt. Der Quarzgenerator besteht aus den Elementen N 10, N 11, N 12, sowie R 1, R 2, C 1, C 2 und natürlich dem Quarz. Ein solcher Quarz ist von der Firma Kristallverarbeitung Neckarbischofsheim GmbH unter der Bezeichnung XS 0803 8 MHz zu einem Preis von ca. 19.80 DM lieferbar. Dieser Takt steuert eine Kette von Synchronzählern Z 1...Z 6 des Typs SN 74 161. Sie haben die Aufgabe, die Adressen für die Speicher und für den Zeichengenerator zu erzeugen. Mit den NAND-Verknüpfungen N 5, N 6 und N 7 werden die Übernahmesteuerimpulse ST 1 und ST 2 gewonnen. FF 1, FF 2 und M 1...M 6 dienen der Erzeugung der Bildsynchronsignale. Jeweils eine Gruppe von Trimmern, P 1...P 3 und P 4...P 6, dienen der Einstellung von Zeilensynchron- und Bildsynchronsignal. Die genauen Einstellungen der Trimmer sind mit Hilfe eines angeschlossenen TV-Gerätes durchzuführen. Dazu schließt man dieses Steuerteil, genau gesagt die Anschlüsse Sy 1 und Sy 2, an den BAS-Mischer. Den Videoeingang des BAS-Mischers kann man unbeschaltet lassen. Der Ausgang des BAS-Mischers wird dann mit dem Eingang der Hf-Stufe verbunden. Dann kann wie folgt ein sauber stehendes TV-Bild eingestellt werden. Mit den Trimmern P 2 und P 5 kann die Lage des Zeichenfeldes bestimmt werden. Beim Abgleich wird am besten mit der Einstellung der Trimmer P 1 und P 3 begonnen. Sie bestimmen die Art des Zeilensynchronsignals. Wenn das Bild in dieser Hinsicht stabil ist, kann durch Einstellen der Trimmer P 4 und P 6 ein stehendes und in bezug auf die Verdunkelung des Strahlrücklaufes, ein sauberes Bild eingestellt werden. Der endgültige Feinabgleich aller Trimmer kann nur erfolgen, wenn Zeichen auf dem Bildschirm sichtbar sind. Denn dann kann auch der Bildkontrast durch Veränderung der Schwarzschulter mit

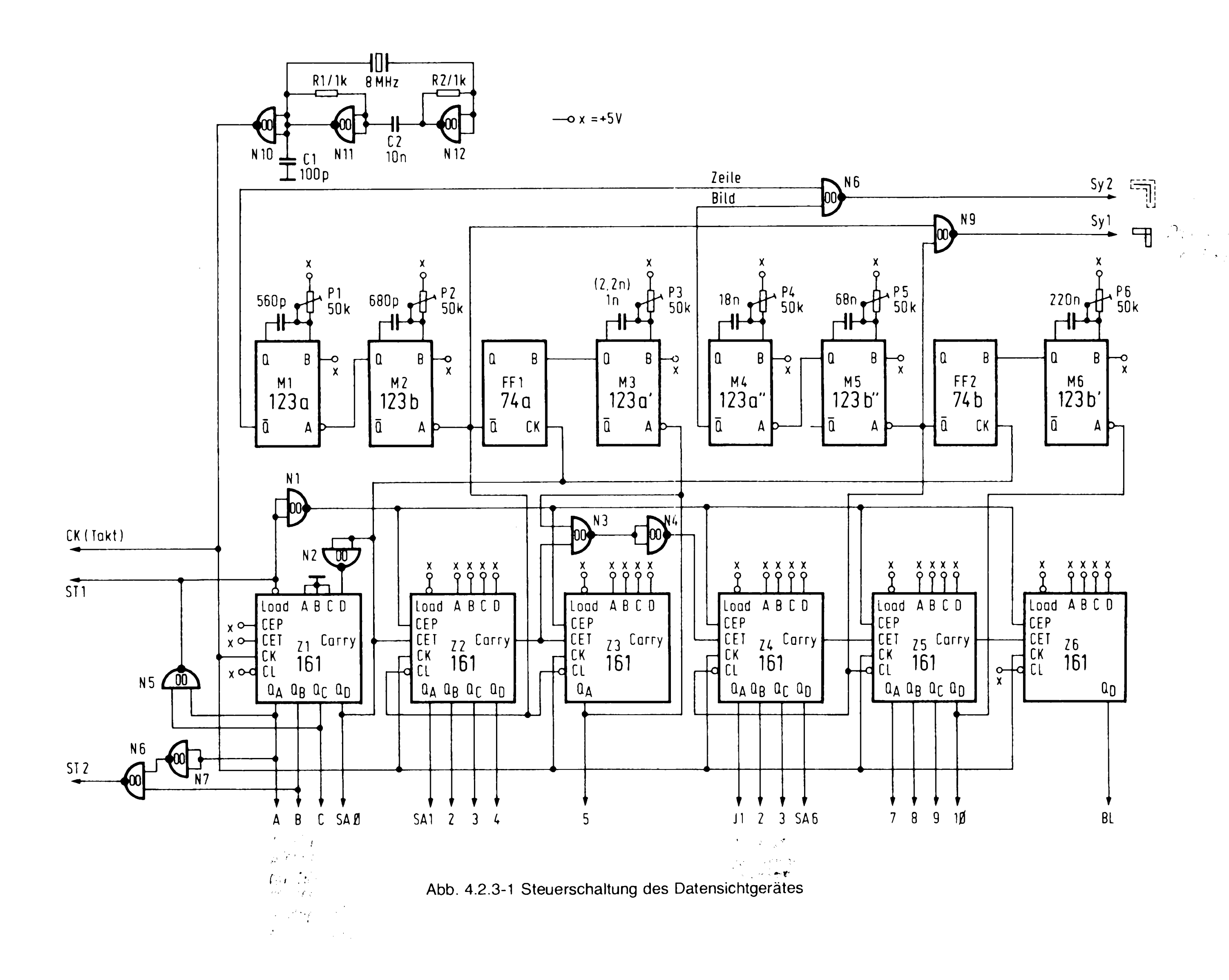


Abb. 4.2.3-1 Steuerschaltung des Datensichtgerätes

P 1 und P 3 noch einmal genau eingestellt werden.

4.2.4 Zeichengenerator

Abb. 4.2.4-1 zeigt die Schaltung des Zeichengeneratorteils. Es wird hier der schon in Abschnitt 4.1.2 erklärte Zeichengenerator TMS 2501NC verwendet. Die Speicher haben eine gewisse Zugriffszeit, die in ihrer Größe der Zeit nahe kommt, die der Strahl des TV-Gerätes braucht, um eine Zeile eines Zeichens abzutasten. Somit würde die Information über das darzustellende Zeichen viel zu spät beim Zeichengenerator ankommen. Daher ist es notwendig, diese Schwierigkeit mit einem Kunstgriff zu umgehen. Eine schon in der Steuereinheit berücksichtigte Verzögerung der Adressen des darzustellenden Zeichens und der tatsächlichen Anzeigeposition des Zeichens auf dem Bildschirm ist die Lösung dieses Problems. Die Synchronsignale werden mit den beiden Flipflops FF 1 und FF 2 der Abb. 4.2.3-1 um eine Zeichenbreite verzögert.

Im Zeichengeneratorteil wird die Information über das darzustellende Zeichen mit Hilfe des Steuersignals ST 2 kurz nach Darstellung der Zeile des letzten Zeichens in den Zwischenspeicher SP 1 übernommen. Es liegt nun die Information an dem Zeichengenerator-IC an. Dieser gibt an seinen fünf Ausgängen die durch den Zustand von J 1, J 2 und J 3 bestimmte Zeile dieses Zeichens aus. Mit ST 1 wird diese Information auch sofort in den zweiten Zwischenspeicher SP 2 übernommen. Die Information kann dann von dem Multiplexer des Typs SN 74151 abgetastet werden. Dem Multiplexer wird dazu die Adresse über die Anschlüsse A, B und C von dem Steuerteil zugeführt. An den Strobeeingang des Multiplexers gelangt noch die Taktfrequenz von 8 MHz. Dadurch wird erreicht, daß zwischen den einzelnen Bildpunkten in horizontaler Richtung jeweils ein kleiner Zwischenraum bleibt. Der Multiplexer besitzt zwei Ausgänge. Einer stellt dabei die invertierte Information des anderen dar.

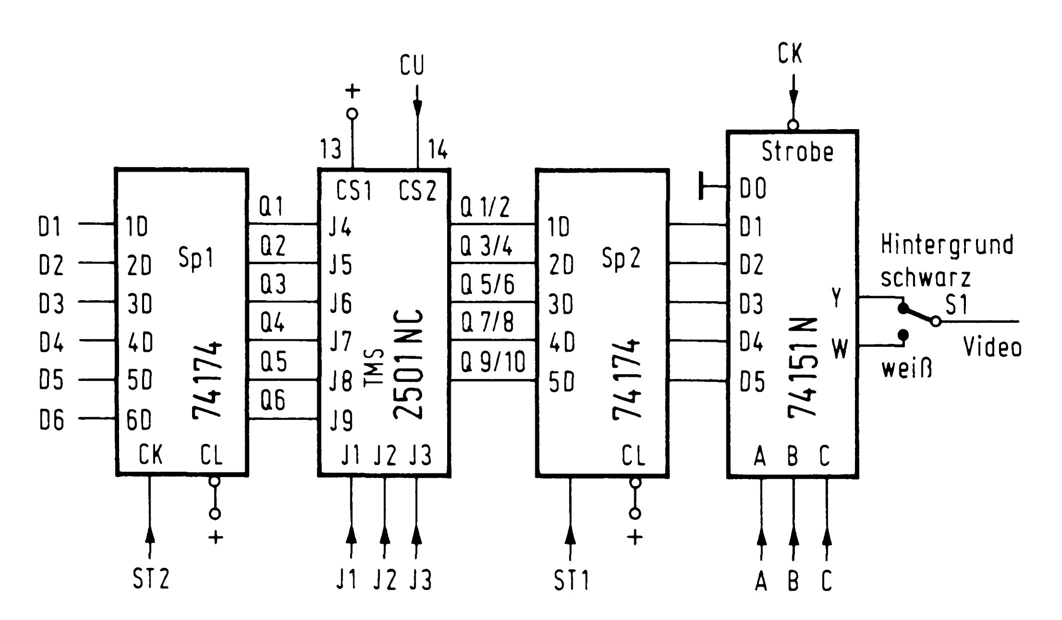


Abb. 4.2.4-1 Zeichengeneratorteil

Es ist somit möglich, falls der im Schaltbild eingezeichnete Schalter S 1 vorgesehen wird, zwischen schwarzer Schrift auf weißem Grund und weißer Schrift auf schwarzem Grund zu wählen.

4.2.5 Speichersteuerteil

Diese Schaltung, die *Abb. 4.2.5-1* zeigt, dient dazu, die Position des nächsten einzuschreibenden Zeichens festzuhalten und die Umschaltung zwischen interner Adresse und dieser Positionsadresse vorzunehmen. Ferner muß diese Schaltung alle Steuerbefehle wie „carriage return", „line feed" ausführen.

Die Speicher für die Zeichenposition bestehen aus vier Vor- und Rückwärtszählern Z 1...Z 4 des Typs SN 74193. Im normalen Fall, wenn ein Zeichen in den Speicher eingeschrieben werden soll, erhöht sich der Inhalt der Zählerkette nach dem Einschreibvorgang um eins. Eine Logik, bestehend aus den NAND-Verknüpfungen N 17...N 20, sorgt dafür, daß nach Zeilenende ein korrekter Übertrag an den folgenden Zähler Z 3 gegeben wird.

Über den Rückwärtszähleingang des Zählers Z 1 ist es ferner möglich, die Positionsadresse um eins zu verringern, das heißt, innerhalb einer Zeile die Position des nächsten einzuschreibenden Zeichens um eins nach links zu versetzen (außer am linken Zeilenrand). Auf ähnliche Art und Weise werden die anderen Befehle möglich gemacht, wie noch anhand von ein paar Beispielen gezeigt wird (andere Befehle sind zum Beispiel: „Home", „carriage return", „line feed", „up", „down", „forward", „clear all", „read only", „write"). Mit dem Befehl „read only" ist es möglich, das Auslesen des Inhalts des Speichers an externe Geräte zu erlauben. Ein LOW-Signal auf dieser Leitung sperrt in dem Speichersteuerteil die Einschreibsignale für den Speicher.

Die Umschaltung zwischen den zwei Adreßquellen für die Zeichenposition übernimmt die Multiplexerstufe, bestehend aus U 1, U 2 und U 3 des Typs SN 74157.

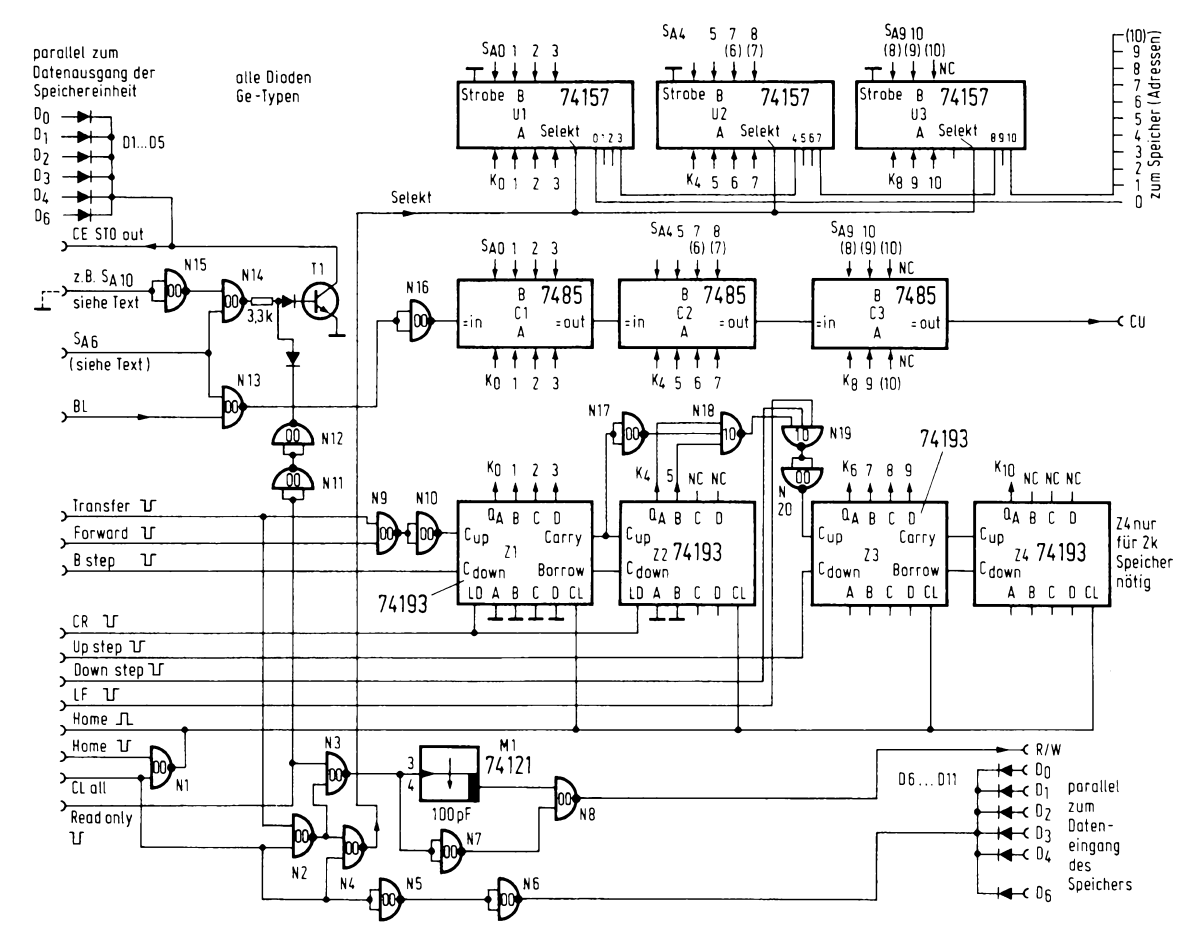
Die Umschaltung wird immer dann vorgenommen, wenn der Pegel aus der sogenannten Transferleitung (Übernahmeleitung) auf LOW-Potential liegt. Bei der Ausführung des „clear"-Befehls wird trotz eines Einschreibvorgangs diese Umschaltung nicht vorgenommen. Dabei erfolgt gleichzeitig die automatische Eingabe des Zeichens „space" (Leerraum) an den Dateneingang des Speichers. Somit wird der Inhalt des Speichers innerhalb einer Zeitdauer von 1/50 s bei anhaltendem „clear"-Signal mit Leerzeichen überschrieben und der Bildschirm erscheint dunkel oder hell, je nach Stellung von S 1.

Das Datensichtgerät ermöglicht die Darstellung eines sogenannten Cursors. Dies ist hier ein blinkendes Feld, das angibt, wo das nächste Zeichen erscheint, das eingegeben wird.

Der Cursor entsteht durch Vergleich der Adresse in den Zählern Z 1...Z 4 mit der Adresse, die von der Steuereinheit gegeben wird. Den Vergleich übernimmt eine Stufe, bestehend aus C 1...C 3 des Typs SN 7485. Dieser Stufe wird ein zusätzliches Signal BL über eine Logik zugeführt. Dieses Signal besitzt einen langsam pulsierenden Pegel. Es wird nun ab Abhängigkeit des Vergleichs an den Ausgang der Zählerstufe geschaltet oder nicht. Dieser Ausgang CU gelangt dann an den Zeichengeneratorteil mit gleichnamigem Eingang.

Die Schaltung ermöglicht, wie schon gesagt, 2048 Zeichen auf dem Bildschirm darzustellen. Dabei leidet aber die Ablesbarkeit der Zeichenfolgen stark. Es ergeben sich zu geringe Zwischenräume zwischen den Zeichen in vertikaler Richtung. Wie schon erwähnt, ist es möglich, den Speicher nur für 1024 Zeichen vorzusehen und dabei die Zeichen so zu verteilen, daß in vertikaler Richtung auf dem Bildschirm jeweils eine Leerzeile zwischen den Zeichenzeilen entsteht.

Die Steuereinheit liefert aber die Adressen


Abb. 4.2.5-1 Speichersteuerschaltung

für alle 2048 Zeichen. Sie liegen an den Ausgängen SA 0...SA 10 des Steuerteils.

Um nun die Darstellung mit den 1024 Zeichen nach obiger Beschreibung zu ermöglichen, ist es nötig, die Adresse SA 6, die ja den Zeilenwechsel an niederwertigster Stelle angibt, einer besonderen Stufe, bestehend aus N 13, N 14, T 1 und D 1...D 5 zuzuführen. Mit der NAND-Verknüpfung N 13 wird erreicht, daß die Darstellung des Cursors nur auf Zeilen erfolgen kann, auf denen auch wirklich Zeichen vorhanden sein können. Mit N 14 und T 1 und den Dioden D 1...D 5 wird erzielt, daß die Zeilen mit Leerzeichen gefüllt erscheinen, die dazwischen liegen.

Wird gewünscht, den vollen Speicherbereich von 2048 Bytes zu benutzen, so wird SA 6 nicht an die Verknüpfungen N 13 und N 14 angeschlossen. Es werden dann die in Klammern auf dem Schaltplan eingezeichneten Adressen an die Stufen U 2 und U 3 sowie an C 2 und C 3 angeschlossen.

Durch die Verwendung von RAM-Speichern im Speicherteil, die eine 256 x 4-Organisation verwenden, ist es möglich, den Speicher nicht sofort auf einmal zu beschaffen,

sondern stufenweise aufzubauen. Nicht vorhandene Adressen können dann notfalls über eine zusätzliche Logik an die NAND-Verknüpfung N 15 angeschlossen werden. Ist zum Beispiel ein Speicherplatz von 512 Bytes verfügbar, so würden, wenn diese Zusatzschaltung nicht da wäre, im unteren oder oberen Bildteil undefinierte Zeichen erscheinen. In diesem Fall wird die Leitung SA 10 zusätzlich an den Eingang N 15 geschaltet. Es wird dadurch erreicht, daß auf dem Bildschirm an den Stellen, an denen kein Speicher vorhanden ist, auf jeden Fall Leerzeichen erscheinen.

4.2.6 Speicher

Als Speicher-IC wurde der Typ 2606 B-1 von Signetics gewählt. Dieser Speicher besitzt eine Kapazität von 1024 bit, die 256 x 4 organisiert sind. Auch ist der Speicher in einem IC mit 16 Anschlüssen untergebracht. Dabei werden die Datenein- und -Ausgaben über eine gemeinsame Bus-Leitung vorgenommen. Wie schon gesagt, ist es durch die 256 x 4 Organisation möglich, den Speicher in Stufen von 256 Bytes aufzubauen. Eine solche Stufe besteht dann immer aus zwei derartigen ICs. Für die eindeutige Bestimmung von 64 darstellbaren Zeichen, wie sie der hier verwendete Zeichengenerator erlaubt, wären eigentlich nur 6-bit-Speicherplätze pro Zeichen notwendig.

Den vorhandenen Speicher dahingehend auszunützen, würde aber einen größeren Aufwand oder eine geringere Flexibilität erzwingen.

So ist es nun aber auch möglich, durch Verwendung eines Zeichengenerators mit 128

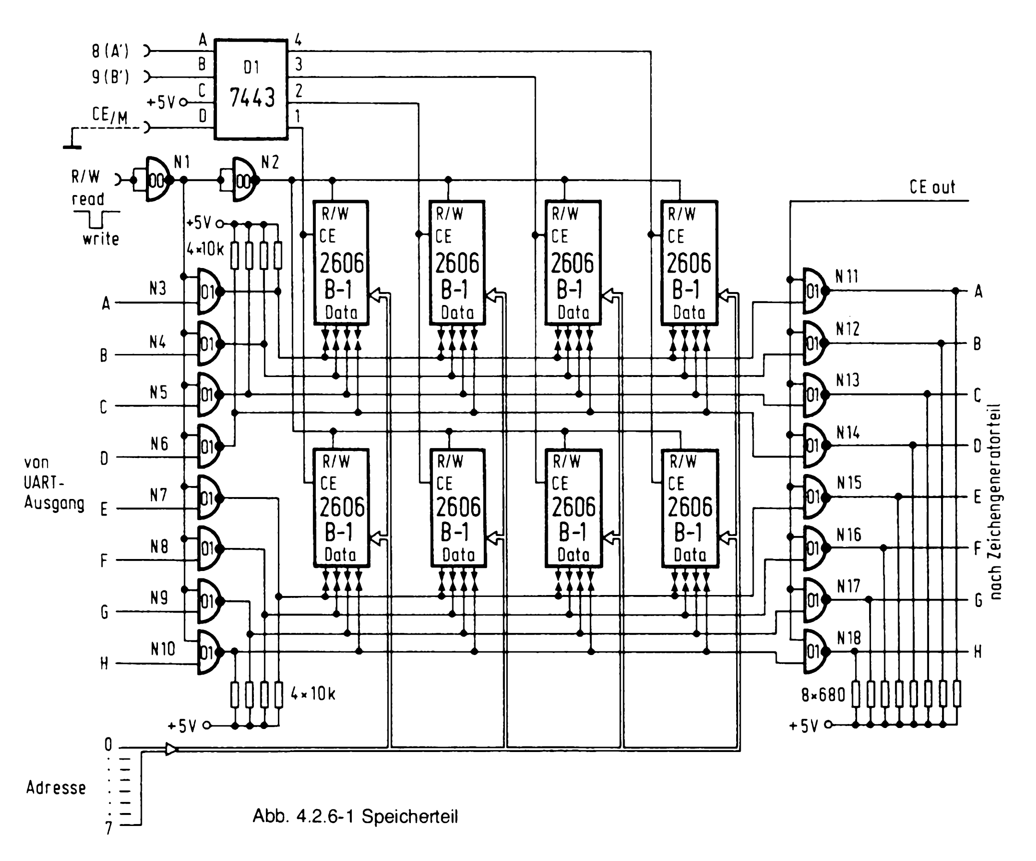


Abb. 4.2.6-1 Speicherteil

darstellbaren Zeichen Groß- und Kleinschreibung durchzuführen, ohne daß eine Veränderung des Speicherteils notwendig wäre. Der Speicher kann auch als Arbeitsspeicher für ein 8-bit-Mikrocomputersystem (ein Mikrocomputer wird noch in einem späteren Kapitel beschrieben) verwendet werden, ohne wesentliche „Umbauten" an der Schaltung vornehmen zu müssen. Als Speicher-ICs können selbstverständlich auch andere Fabrikate eingesetzt werden, falls sie eine vergleichbare Zugriffszeit besitzen. Da der Speicher durch die 256-Organisation nicht direkt eindeutig adressiert werden kann, ist es nötig, eine zusätzliche Decodierschaltung anzubringen. Sie besteht aus dem IC D 3 des Typs SN 7443, der aufgrund seiner besonderen Struktur eine einfache Erweiterung des Speichers auf 2048 Bytes zuläßt.

Dazu wird die Speicherschaltung zweimal aufgebaut, und beim Zusammenschalten der beiden Speicher werden alle Eingänge der beiden Schaltungen verbunden bis auf den Eingang CE/M des Decoders.

Bei der einen Schaltung wird dieser, wie gestrichelt eingezeichnet, mit Masse verbunden. Dafür wird der Eingang CE/M der anderen Speicherschaltung nicht mit Masse verbunden, sondern mit dem Eingang C des Decodierers der ersten Speicherschaltung. Dieser Eingang C, der im Schaltbild an + 5 V angeschlossen ist, wird nun nicht an dieses Potential geschaltet, sondern bildet einen neuen Adreßeingang. Die Ausgänge der Speicherschaltungen werden vor den Verknüpfungen N 11...N 18 miteinander verbunden. Diese Stufe braucht dann nur einmal aufgebaut zu werden. Würde man die Ausgänge hinter dieser Stufe miteinander verbinden, so ergäben sich Störungen, die zu einem Nicht-Funktionieren der Schaltung führten. Über die NAND-Verknüpfung N 1 gelangt der Lese-Schreibbefehl an die Stufen N 3...N 10. Jedesmal bei einem Schreibbefehl werden die Eingangsdaten an den Speicher weitergegeben.

Über N 2 gelangt der Befehl an den eigentlichen Speicherteil. Mit Hilfe des Eingangs CE/OUT können die Ausgänge der NAND-Verknüpfungen N 11...N 18 in den „open collector"-Zustand überführt werden. Der Pegel liegt dann aufgrund der Widerstände auf einem definierten HIGH-Zustand.

Dieser Eingang wird von der Speichersteuereinheit verwendet, um Leerzeilen zwischen zwei Zeichenzeilen, wie schon beschrieben, zu erzeugen.

Es gelangt nämlich gleichzeitig dieses Signal, das auch an den Eingang CE/OUT führt, an eine Reihe von Dioden in der Speichersteuereinheit, die das Zeichen „space" (Leerraum) codieren.

4.2.7 Hilfsschaltung für Tastatur

Diese Hilfsschaltung kann bei der Verwendung bestimmter Tastaturen notwendig werden. *Abb. 4.2.7-1* zeigt die Schaltung. Sie wird benötigt, wenn Tastaturen verwendet werden, die keine Möglichkeit haben, Sonderbefehle wie „carriage return", „escape"... einzugeben. Die Schaltung wurde beim Aufbau mit einer Tastatur von Rafi Typ 3.91102.001 verwendet. Diese Tastatur liefert die Information im ISO-7-bit-Code für Groß- und Kleinbuchstaben.

Zur Realisierung der Sonderbefehle, wie für eine zusätzliche Zehnertastatur, benötigt man noch ein kleines Tastenfeld. Eine günstige Lösung bietet z. B. das von Rafi angebotene Tastenfeld mit der Typenbezeichnung 3.81103.001. Es verwendet entprellte Tasten und enthält keine Decodierschaltung. Dieses Tastenfeld besitzt 14 Tasten, von denen 10 mit den Ziffern 0...9 beschriftet sind, und 4 zur freien Verfügung stehen. Die im Bild dargestellte Schaltung gestattet die Codierung dieser Zifferntasten, sowie die Codierung zweier Sonderbefehlstasten, „carriage return" und „escape". Die beiden Tasten des Tastenfeldes,

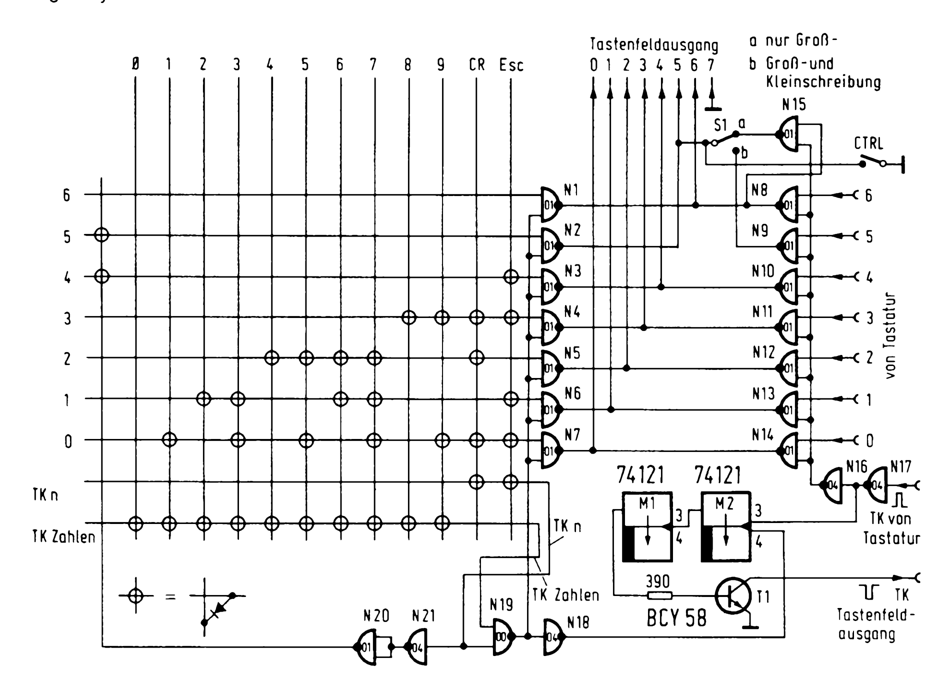


Abb. 4.2.7-1 Interface für Tastatur

die nicht codiert werden, können z. B. direkt mit den Eingängen der Speichersteuerschaltung verbunden werden, die die Bezeichnung Forward und Bstep tragen. Somit ist es möglich, die Position des Cursors zu verändern, ohne über die Datenleitungen operieren zu müssen.

Die Schaltung enthält zwei weitere Tasten bzw. Schalter. Mit Schalter S 1 ist es möglich, zu bestimmen, ob nur Großbuchstaben ausgegeben werden sollen, oder aber Groß- und Kleinbuchstaben, falls dies die beim Nachbau verwendete Tastatur zuläßt. Mit der Taste CTRL ist es möglich, alle weiteren Sonderbefehle zu codieren. Dazu wird diese Taste gleichzeitig mit einer Taste der alphanumerischen Tastatur betätigt. *Tabelle 4.2.7-1* zeigt eine Gegenüberstellung der Befehle, die das Datensichtgerät ausführen kann, und der Tastenkombination, die den dazugehörigen Code erzeugt. Näheres über das Zustandekommen

Tabelle 4.2.7-1

Befehl beim Datensichtgerät	CTRL ⊕ Zeichen auf Tastatur
frei	Space
frei	!
frei	”
frei	#
frei	$
Home	%
frei	&
Cear all	’
Rückwärtsschritt ←	(
Vorwärtsschritt →	)
Zeilenvorschub ↓	*
nach oben ↑	+
frei	,
Wagenrücklauf ↓	–
Read on	.
Read off	/

der Befehle wird im Abschnitt 4.2.9 noch genannt.

Die Codierung der Tasten des Tastenfeldes erfolgt über eine speziell geartete Diodenmatrix. Speziell deshalb, weil für die Ziffern eine Vereinfachung des Codierungsvorgangs gegeben ist. Es stehen zwei gesonderte Hilfsleitungen zur Verfügung. Eine Leitung trägt die Bezeichnung TKn und eine TK-Zahl. Die Ziffern haben in der Codierung gemäß dem ISO-7-bit-Code in dem Bitteil 4 und 5 genau die gleiche Informationsfolge. Es wäre nun in dieser Schaltung umständlich gewesen bzw. aufwendiger, den sich wiederholenden Bitteil in der Diodenmatrix in Form von Dioden auch zu wiederholen. Für den eigentlichen Codierungsvorgang stehen Leitungen mit der Numerierung 0...6 zur Verfügung. Für ein beliebiges Zeichen, das im ISO-7-bit-Code codiert wird, wird wie folgt verfahren. Es wird immer dort eine Diode, wie im Schaltbild angegeben, in die Diodenmatrix in die Leitung eingesetzt, die beim ISO-7-bit-Code des betreffenden Zeichens ein Signal 1 haben muß. Ferner muß eine Diode an die Leitung TKn angeschlossen werden. Dadurch wird ein Takt bzw. Übernahmeimpuls ausgelöst, wenn die betreffende Taste betätigt wird und somit die Leitung TKn auf ein LOW-Pegel legt.

Werden Zahlen codiert, so werden nur die Bitstellen 0...3 im BCD-Code codiert und eine Diode wird an die Leitung TK-Zahl angeschlossen. Eine Codierung im ISO-7-bit-Code wird dann automatisch ausgegeben. Die Monoflops M 1 und M 2 dienen der Erzeugung des Übernahmeimpulses. Dieser Übernahmeimpuls wird dann dem UART zugeführt (dem seriellen Interfaceteil). Am Ausgang des Transistors T 1 liegt dieses Signal an. Die Daten der Tastatur gelangen immer dann über die NAND-Verknüpfungen N 8...N 14 an den Ausgang der Schaltung, wenn der Takt, der von der Tastatur geliefert wird, auf HIGH-Potential liegt.

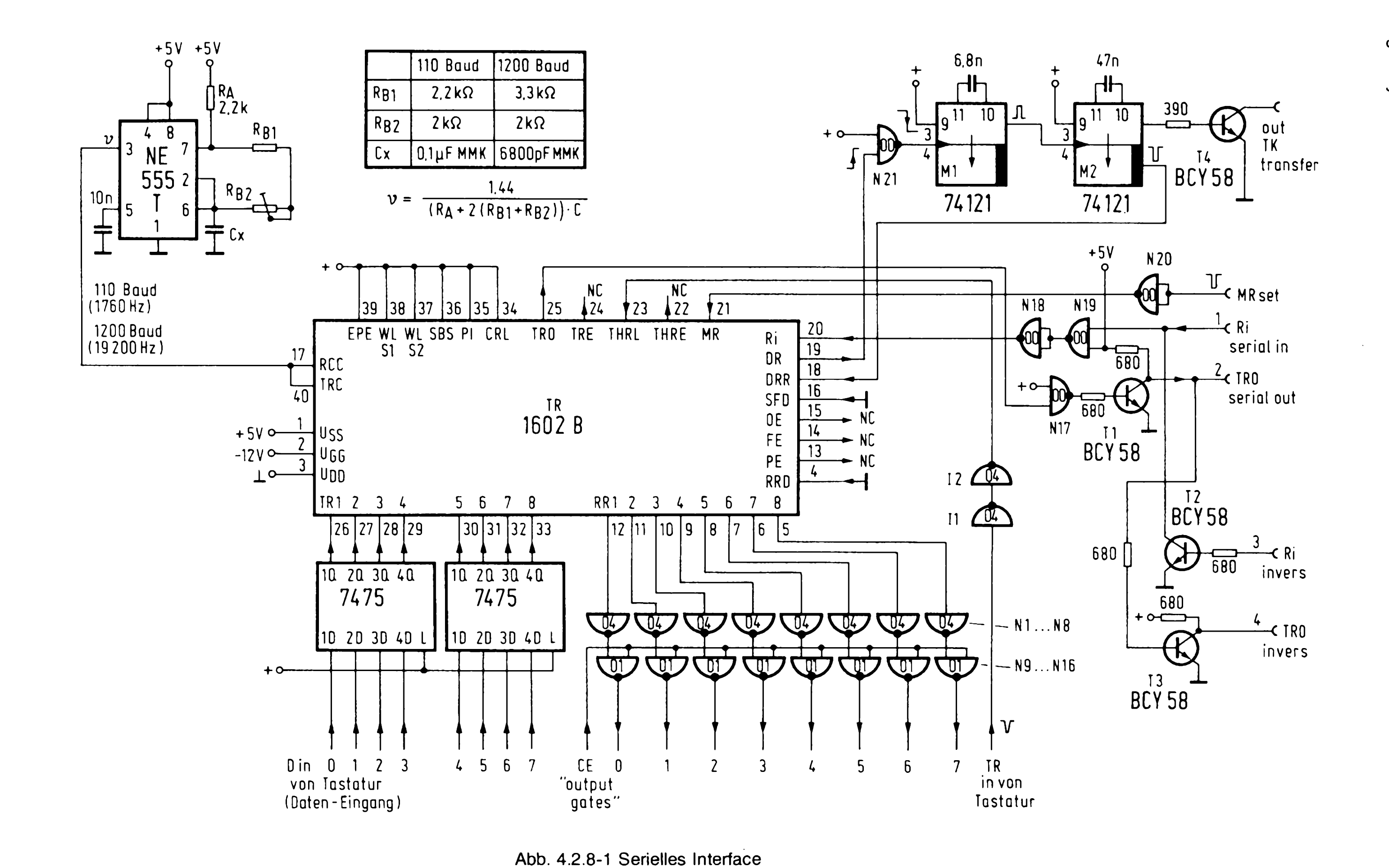
Bei Anstieg auf das Potential HIGH wird gleichzeitig das Monoflop M 2 getriggert, und wenn an dessen Ausgang das Signal nach einer Zeit wieder auf LOW abfällt, triggert das Monoflop M 1, das nun den Übernahmeimpuls an den Ausgang TK liefert. Bei Betätigung einer der Sondertasten werden die Daten über N 1...N 7 auf den Ausgang der Hilfsschaltung geschaltet und gleichzeitig das Monoflop M 2 in gleicher Weise getriggert. Dies führt dann zu einem gleichgearteten Taktimpuls am Ausgang TK.

4.2.8 Interface für serielle Datenübertragung (UART)

Diese Schaltung ermöglicht z. B. dem Benutzer des Datensichtgerätes, über eine serielle Schnittstelle ein Mikrocomputersystem direkt anzuschließen.

Abb. 7.2.2-1b des Kapitels 7 zeigt das Signal, wie es am Ausgang der Schaltung erscheint (oder als Eingangssignal der Schaltung zugeführt werden kann). Der im Bild eingezeichnete Übergang des Signals von HIGH auf LOW triggert sozusagen die UART-Schaltung. Dann folgen nach einer kurzen definierten Verweilzeit die Daten, und nach den Datenbits folgen noch zwei Stoppbits, die das Ende der Übertragung eines Zeichens angeben.

Die integrierte Schaltung des Typs TR 1602 B (lieferbar von der Firma Spezial Elektronik) beinhaltet ein komplettes Datenübertragungssystem für eine solche Übertragung (sogar für Senden und Empfangen). Die Übertragungsgeschwindigkeit wird in BAUD angegeben. Die normale Übertragungsgeschwindigkeit für einen Fernschreiber, wie er für Kommunikation mit einem Mikrocomputer verwendet wird, beträgt 110 BAUD. Dies entspricht einer Zeichenübertragungsgeschwindigkeit von ungefähr 10 Zeichen in der Sekunde. Zur Erzeugung der Taktfrequenz ist ein Taktgenerator mit einer relativ stabilen Frequenz nötig.



	110 Baud	1200 Baud
R_{B1}	2,2 kΩ	3,3 kΩ
R_{B2}	2 kΩ	2 kΩ
Cx	0,1 µF MMK	6800 pF MMK

$$\nu = \frac{1{,}44}{(R_A + 2(R_{B1} + R_{B2})) \cdot C}$$

Abb. 4.2.8-1 Serielles Interface

Die Erzeugung des Taktes wird mit Hilfe des ICs NE 555 vorgenommen, das in bezug auf Stabilität den Forderungen entspricht. In *Abb. 4.2.8-1*, die die Schaltung zeigt, ist eine kleine Tabelle angegeben, die die Dimensionierung der frequenzbestimmenden Bauteile für zwei verschiedene Baud-Raten zeigt. Einmal wurde eine Dimensionierung für die gebräuchliche Rate von 110 Baud angegeben und dann noch die Dimensionierung für eine Rate von 1200 Baud, die bei manchen Mikrocomputerbausätzen (z. B. dem SDK 80 Kit) einstellbar ist und einen komfortableren Betrieb ermöglicht.

Bei der Auswahl des angegebenen Kondensators Cx ist auf die Verwendung eines qualitativ guten Typs zu achten, da sich sonst nicht die erforderliche Stabilität ergibt. Der Trimmer RB 2 muß noch abgeglichen werden. Dies geschieht am besten mit Hilfe eines Frequenzmeßgerätes. Die Ausgangsfrequenz des Taktgenerators ergibt sich aus der Multiplikation der gewählten Baud-Rate mit dem Wert 16.

Die Schaltung sieht zwei Ein- und Ausgänge vor, die jeweils invertierte Pegel annehmen oder abgeben. Bei dem Eingang „RI serial in" muß das eingegebene UART-Signal als Ruhezustand den HIGH-Pegel annehmen, bei dem Eingang „RI invers" entsprechend umgekehrt. Der Ausgang „TRO serial out" liefert ein UART-Signal mit HIGH als Ruhelage und der Ausgang „TRO invers" entsprechend umgekehrt.

Durch die beiden Ein- und Ausgänge ist eine Kompatibilität mit verschiedenen Systemen gewährleistet. Der jeweilige Ein- und Ausgang kann dann entsprechend gewählt werden.

4.2.9 Befehlsdecodierteil

Abb. 4.2.9-1 zeigt das Schaltbild dieses Teils des Datensichtgerätes.

Diese Schaltung hat im wesentlichen die Aufgabe, die ankommenden Daten gemäß *Tabelle 4.2.9-1* zu entschlüsseln. Dabei wird zusätzlich dafür gesorgt, daß solche Steuerbefehle nicht in den Datenspeicher eingeschrieben werden.

Tabelle 4.2.9-1

Code	Bedeutung im ISO-7-bit-Code	Wirkung im Datensichtgerät
00_{16}	NUL Füllzeichen	frei benutzbar
01_{16}	SOH Anfang des Kopfes	frei benutzbar
02_{16}	STX Anfang des Textes	frei benutzbar
03_{16}	ETX Ende des Textes	frei benutzbar
04_{16}	EOT Ende der Übertragung	frei benutzbar
05_{16}	ENQ Stationsaufforderung	„Home"
06_{16}	ACK Positive Rückmeldung	frei benutzbar
07_{16}	BEL Klingel	„clear all"
08_{16}	BS Rückwärtsschritt	Rückwärtsschritt (left)
09_{16}	HT Horizontal-Tabulator	Vorwärtsschritt (right)
$0A_{16}$	LF Zeilenvorschub	Zeilenvorschub (down)
$0B_{16}$	VT Vertikaltabulator	nach oben (up)
$0C_{16}$	FF Formularvorschub	frei benutzbar
$0D_{16}$	CR Wagenrücklauf	Wagenrücklauf
$0E_{16}$	SO Dauerumschaltung	Daten ausgeben „read on"
$0F_{16}$	SI Rückschaltung	Normalbetrieb

Soll der Speicher zum Beispiel als Programmspeicher verwendet werden und so alle 8 bit benötigen, so ist es nötig, die Decodierschaltung, die aus den Decodierern D 1 und D 2 besteht, außer Betrieb zu setzen. Dies geschieht durch Auftrennen der Verbindung des Ausgangs der NAND-Verknüpfung N 1, die zum Eingang der Verknüpfung N 6 und O 1 führt.

Die Schaltung des Datensichtgerätes besitzt einen ergänzenden Peripherie-Ein- und -Aus-

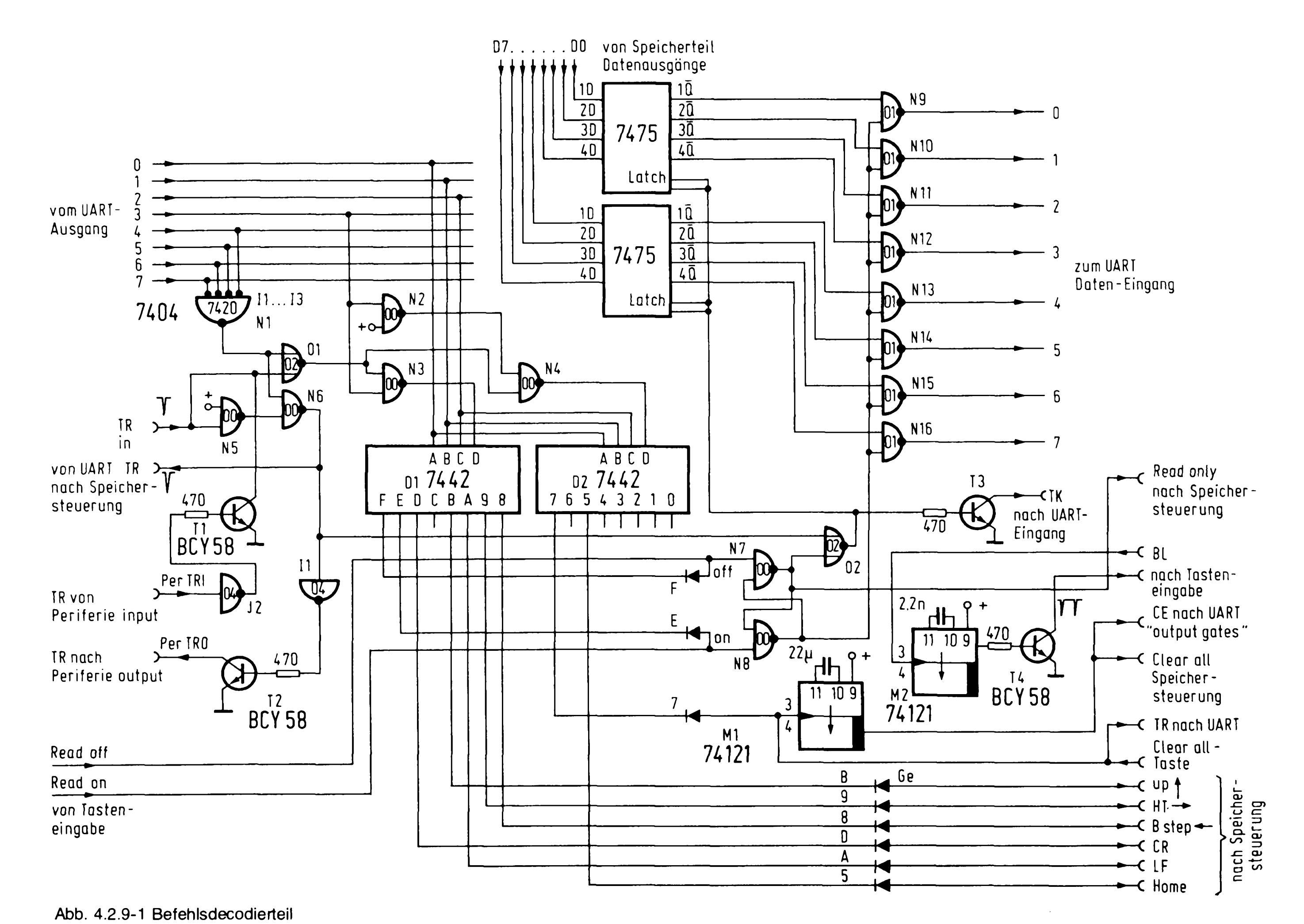


Abb. 4.2.9-1 Befehlsdecodierteil

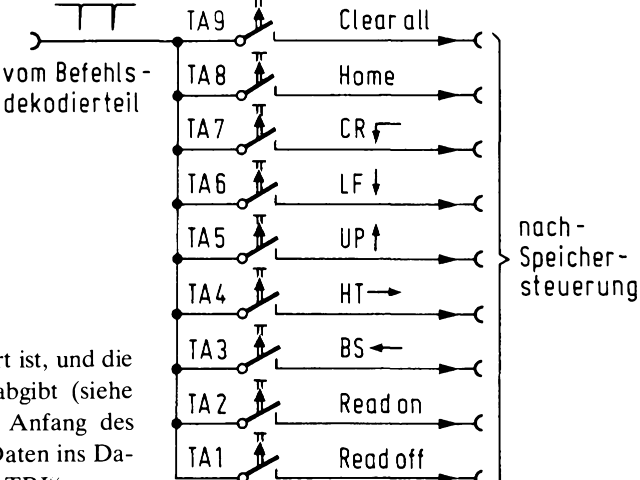


Abb. 4.2.9-2 Tastenfeld für manuelle Eingabe der Steuerbefehle

gang, der als Datenbus ausgeführt ist, und die Daten parallel empfängt bzw. abgibt (siehe Blockschaltung Abb. 4.2-1 am Anfang des Kapitels). Zur Übernahme der Daten ins Datensichtgerät ist der Eingang „Per TRI" vorgesehen. Erfolgt ein LOW-Impuls an diesem Eingang, so werden die am Datenbus anstehenden Daten vom Datensichtgerät weiterverarbeitet. Erfolgt eine Eingabe in das Datensichtgerät über das serielle Interface, so erscheint ein Signal am Ausgang „Per TRO". Damit ist es möglich, die eingegebene Information an ein externes Gerät weiterzugeben.

Übrigens muß der Inhalt des UART den Wert FF_{16} haben, wenn über den Bus Daten in das Datensichtgerät eingegeben werden sollen.

Die Decodierung geschieht, wie schon gesagt, mit den Decodern D 1 und D 2. Man sieht, daß nicht alle Ausgänge belegt sind. Es ist somit dem Anwender möglich, die freien Stellen selbst noch zu belegen. So kann z. B. ein Peripheriegerät damit angesteuert werden (Drucker etc.). Eine Erklärung der schon vorhandenen Befehle erübrigt sich bis auf zwei besondere Befehle.

Es sind dies die Befehle mit der Bezeichnung „read on" und „read off" (Normal-Betrieb).

Doch vor der Erklärung dieser Befehle noch etwas anderes. *Abb. 4.2.9-2* zeigt eine Zusatzschaltung, die aus einer Tastenreihe besteht. Diese Tasten TA 1...TA 9 ermöglichen die manuelle Ausführung von Steuerbefehlen. Dabei werden als Tasten gewöhnliche, nicht entprellte Typen verwendet. Die Entprellung geschieht über ein spezielles Abtastverfahren. Dabei wird in zeitlichen Abständen ein kurzer Impuls an die gemeinsame Leitung der Tasten geführt. Es erfolgt dann die Ausführung der Bedeutung der Tasten entsprechend periodisch mit diesen zeitlichen Abständen. Doch zurück zu der Erklärung der Befehle „read on" und „read off", die nun besser möglich ist.

Wird die Taste „read on" betätigt oder ein entsprechender Steuerbefehl gegeben, so erfolgt eine interne Umschaltung, die die Ausgabe von Informationen aus dem Speicher ermöglicht. Dabei wird über den seriellen Ausgang immer dann ein Zeichen ausgegeben, wenn über den seriellen Eingang ein Zeichen eingegeben wurde, das jedoch kein Steuerzeichen darstellt. Soll der Speicher des Datensichtgerätes als Programmspeicher für einen Mikrocomputer verwendet werden, so muß über einen zusätzlichen Eingang am Datensichtgerät die Umschaltung von „read off" auf „read on" und umgekehrt möglich sein, da ja die interne Decodierung dann entfällt. Die Umschaltung von „read on" auf Normalbetrieb kann nun wieder entweder manuell oder durch einen Steuerbefehl erfolgen.

Wird der serielle Eingang des Datensichtgerätes mit seinem seriellen Ausgang verbunden,

so wird nach Umschaltung auf „read on" und nach Betätigen einer Taste der alphanumerischen Tastatur der gesamte Speicherinhalt automatisch periodisch ausgegeben. Dies gestattet zum Beispiel bei Verwendung eines Modems, das an den seriellen Ausgang angeschlossen wurde, die Aufzeichnung der Daten auf ein Tonband.

Das gesamte Datensichtgerät kann auf 6...7 Europakarten untergebracht werden. Die Europakarten und die Tastatur werden am elegantesten in ein Pultgehäuse eingebaut, wie es z. B. von der Firma Knürr erhältlich ist.

Der Aufbau der Schaltung ist relativ unkritisch. Es ist aber auf eine saubere Masseführung und eine sorgfältige Entstörung der einzelnen ICs mit entsprechenden Kondensatoren zu achten. Die Überprüfung der Schaltung erfolgt am besten stufenweise, indem sie entsprechend der Beschreibung des Datensichtgerätes, wie sie hier gegeben wurde, schrittweise aufgebaut und getestet wird.

5 Drucker

Für eine Datenverarbeitungsanlage stellt der Drucker, sei es Schnelldrucker oder Fernschreiber, einen der wichtigsten Bestandteile dar. Denn ohne Dokumentation sind Programm und Daten meist wertlos.

Auch für einen Mikrocomputer ist ein Drucker ein zwar nicht notwendiges, aber doch recht nützliches Peripheriegerät.

Der hier beschriebene Drucker (Typ Printinadruckteil für OEM-Anwendungen von der Firma Ziegler lieferbar) arbeitet mit einem recht modernen Druckverfahren. Als Druckmedium wird ein metallbeschichtetes Papier verwendet. Diese Beschichtung hat die Eigenschaft, wenn sie unter Spannung gelegt wird, an den Berührpunkten schwarz zu werden. Diese Eigenschaft wird nun ausgenützt, um Information auf diesem Papier festzuhalten.

Es ist dann möglich, Daten auszudrucken, ohne ein Farbband zu verwenden, und der mechanische Aufwand wird außerdem gering gehalten.

Es gibt auch noch andere Druckverfahren, die kein Farbband benötigen, z. B. das Thermodruckverfahren. Dort ist aber das Papier recht teuer.

5.1 Prinzip und Aufbau des Druckers

Abb. 5.1-1 zeigt den schematischen Aufbau des eigentlichen Druckerteils und den Druckvorgang. Die Daten werden bei dem beschriebenen Drucker mit einem 5 x 7-Raster ausgedruckt (ähnliche Darstellung der Zeichen wie in Kapitel 4.1.2 gezeigt). Der eigentliche Druck geschieht mit Hilfe eines Druckkopfes, der mit sieben Elektroden ausgerüstet ist. Somit ist es möglich, wie in Abb. 5.1-1 dargestellt, durch eine Horizontalbewegung alle Punkte zu überstreichen und durch entsprechende Pegel an den Elektroden zu einer bestimmten Zeit die Zeichen auszudrucken. Nach einem anschließenden Rücklauf des Druckkopfes und einem Papiervorschub (bei diesem Drucker nach unten), kann der nächste Druck durchgeführt werden. *Abb. 5.1-2* zeigt schematisch den elektrischen Teil des Druckers. Die Elektroden des Druckkopfes werden über eine Treiberstufe von einem Zeichengenerator angesteuert. Er hat die Aufgabe, die vom Multiplexer ankommende Information im

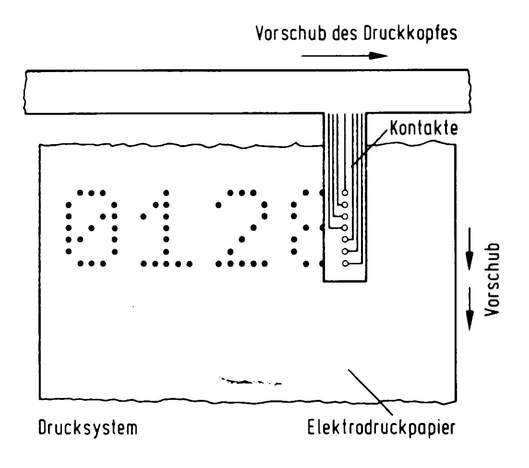


Abb. 5.1-1 Druckprinzip

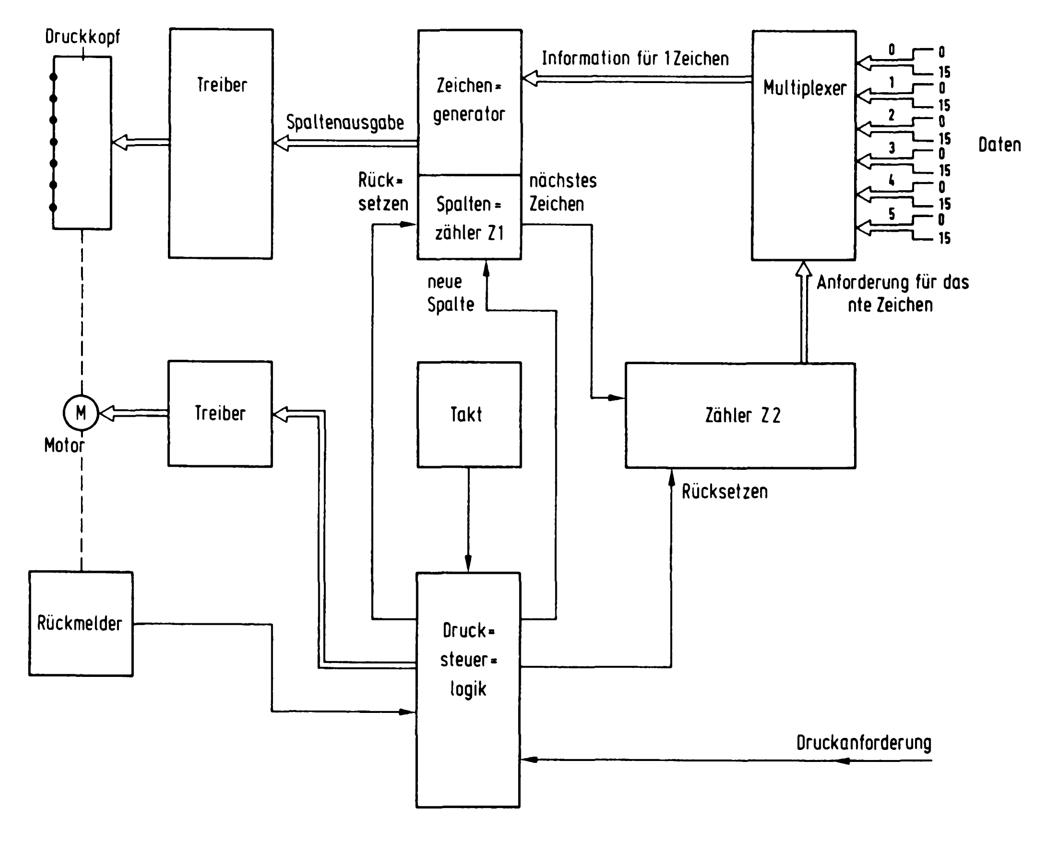


Abb. 5.1-2 Schematische Schaltung des Druckers

ISO-7-bit-Code in die entsprechende Information für das 5 x 7-Raster umzuformen.

Ein im Zeichengenerator vorhandener Spaltenzähler hält die Spalte, über der sich gerade der Druckkopf befindet, fest. Der Zähler bestimmt dann damit die Signale, die an den Druckkopf zur korrekten Darstellung des Zeichens gegeben werden müssen. Der Multiplexer hat die Aufgabe, zu bestimmen, welche der 16 Eingangsdaten gerade bei der jetzigen Position des Druckkopfes gedruckt wird. Er muß die 6 bit dieser Information an den Zeichengenerator weitergeben. Dazu erhält der Multiplexer eine Adresse, die vom Zähler Z 2 geliefert wird. Die Koordination aller Vorgänge übernimmt eine Steuerlogik. Sie bestimmt auch, wann der Druckmotor gestartet wird, und sie bestimmt dann auch die entsprechende Steuerung der Zähler.

5.2 Speicher (mit Schieberegister)

Bei dem hier erklärten Drucker (OEM-Version des Druckers Printina) ist es notwendig, einen externen Datenspeicher zur Verfügung zu stellen. Die Daten für eine vollständige Druckzeile (16 x 6 bit) müssen während des Druckvorgangs statisch und parallel vorhanden sein. Um diese Bedingung zu erfüllen, ist die Verwendung eines Schieberegisters am einfachsten. Dabei müssen die verwendeten Schieberegister parallele Ausgänge für die gespeicherten bit-Stellen besitzen. Dies ist z. B. bei dem Schieberegister vom Typ SN 74164

gewährleistet. Zum Aufbau eines vollständigen Datenspeichers für das beschriebene Druckwerk wären dann 12 ICs nötig (ein IC SN 74164 enthält ein 8-bit-Schieberegister).

Abb. 5.2-1 zeigt den schematischen Aufbau eines solchen Speichers. Zunächst werden die Daten in das Schieberegister eingegeben. Dann, wenn alle 16 Zeichen eingegeben sind, kann der Druckvorgang durch einen auf die Steuerleitung „drucke" gegebenen Impuls den Druckvorgang auslösen. Während des Druckvorgangs dürfen natürlich keine Zeichen in das Schieberegister eingegeben werden.

Um dies zu verhindern, muß entweder mit Hilfe einer Zeitkonstante die hardwaremäßig oder softwaremäßig erzeugt werden kann, gearbeitet werden, oder es muß eine Rückmeldeleitung vorgesehen werden, die angibt, wann der Druckvorgang beendet ist. Bei Verwendung einer Rückmeldeleitung ist eine höchstmögliche Arbeitsgeschwindigkeit erzielbar.

Abb. 5.2-2 zeigt das allgemeine Blockschaltbild für einen derartigen Drucker.

Der Drucker kann zum Beispiel in Verbindung mit dem im vorhergehenden Kapitel beschriebenen Datensichtgerät verwendet werden. In dem Blockschaltbild sind die dafür notwendigen Leitungen bezeichnet. Man benötigt den Peripheriebusausgang des Datensichtgerätes. Als Übernahmebefehl in den Drucker dient die Taktleitung mit der Bezeichnung „Per TRO" am Datensichtgerät. Der Druckvorgang muß nun auch noch irgendwie gesteuert werden. Wenn man sich erinnert, so weiß man, daß noch frei verfügbare Befehle am Ausgang des Befehlsdecoders zur Verfügung stehen. Man kann z. B. den Befehl mit dem Code 03_{16} dazu bestimmen und die Leitung aus dem Datensichtgerät heraus an den Drucker führen.

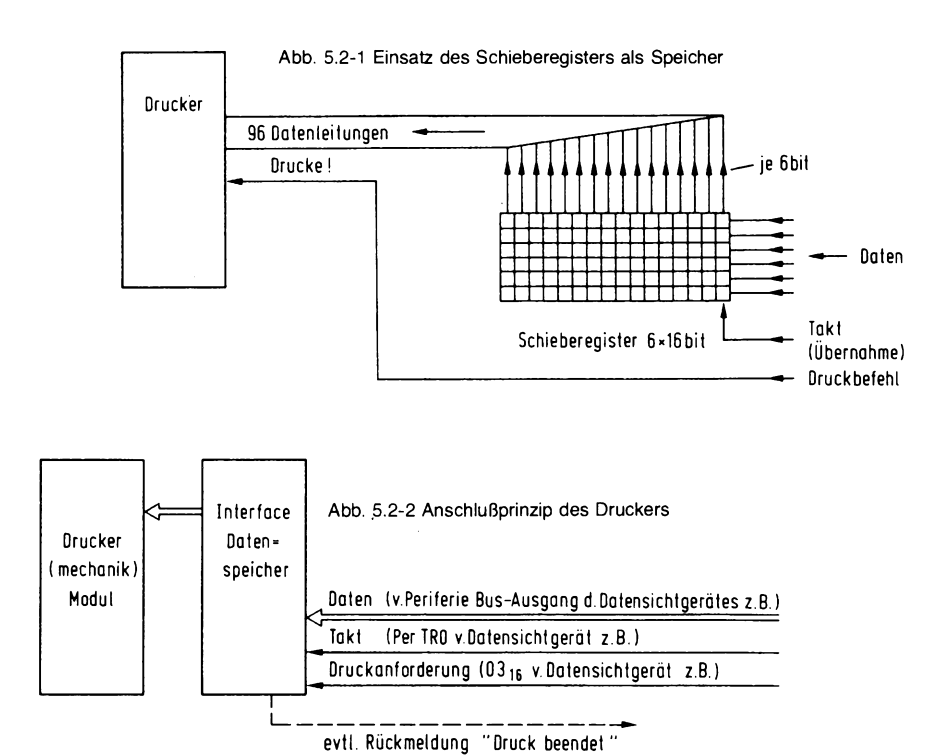


Abb. 5.2-1 Einsatz des Schieberegisters als Speicher

Abb. 5.2-2 Anschlußprinzip des Druckers

5.3 Steuerteil für Bus-Interface

Soll der Drucker zum Beispiel an einen Datenbus, wie er vom Mikrocomputersystem 8080 (über das später noch gesprochen wird), angeschlossen werden, so ist noch ein weiteres zusätzliches Interface notwendig.

Abb. 5.3-1 zeigt das dazugehörige Blockschaltbild. Es soll die Möglichkeit bestehen, den Drucker vom Mikrocomputer aus direkt softwaremäßig anzusprechen. Dazu erhalten die Befehle „Zeichen einlesen“ und „Drucke“ bestimmte Peripherieadressen zugeordnet. Der hier genannte Mikrocomputer besitzt die Möglichkeit, Speicher- und Peripherieadressen zu unterscheiden. Er kann dabei 256 Peripherieadressen und 65536 verschiedene Speicheradressen anwählen. Dabei können auch Speicherbereiche zu Peripherieadressen ernannt werden. Die Trennung erfolgt nämlich durch zwei getrennte Taktleitungen.

Die Zuordnung der Adresse zum entsprechenden Befehl übernimmt ein Decodierer.

Die 8 Datenleitungen werden nicht direkt dem Drucker zugeführt. Der hier dargestellte Drucker hat die Zeichen zwar gemäß dem ISO-7-bit-Code codiert, doch ermöglicht er nur die Darstellung von 64 verschiedenen Zeichen. Er besitzt dafür 6 Eingangsleitungen. Mit einer kleinen Hilfsschaltung ist es möglich, eingegebene Kleinbuchstaben automatisch in den Code für entsprechende Großbuchstaben umzuformen. Dies geschieht ganz einfach durch die Verwendung eines Inverters, der wie im Schaltbild eingezeichnet, in die Datenleitung zum Drucker hin eingefügt wird.

Eine Rückmeldeleitung vom Drucker wäre im Falle der hier gezeigten Anwendung besonders nützlich, da sie eine koordinierte Abfolge der Befehle ermöglicht, besonders bei Arbeiten mit maximaler Druckgeschwindigkeit.

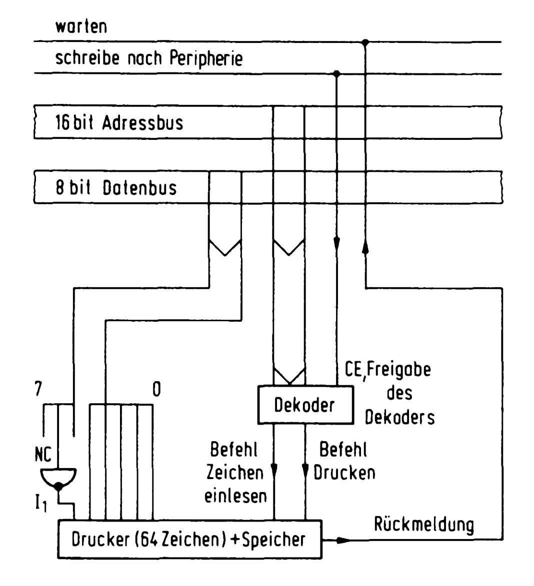


Abb. 5.3-1 Verbinden des Druckers mit einem Datenbus

6 Eingabesysteme

Hier sollen nun verschiedene Möglichkeiten zur manuellen Eingabe von Informationen gezeigt werden, sowie deren technische Realisierung.

6.1 Einzelne Tasten für die Eingabe

6.1.1 Verschiedene Tastenarten

Reed-Tasten

Diese Tastenart arbeitet mit sogenannten Reed-Kontakten. Wenn ein solcher Reed-Kontakt einem Magnetfeld in geeigneter Weise ausgesetzt wird, so schließt der Reed-Kontakt (oder öffnet, je nach Art des Reed-Kontaktes). Verringert sich der magnetische Fluß wieder entsprechend, so öffnet sich der Kontakt wieder (oder schließt). Man kann nun einen solchen Reed-Kontakt in ein Tastengehäuse einbauen. *Abb. 6.1.1-1* zeigt den prinzipiellen Aufbau eines solchen Reed-Tasters. Wenn der Taster betätigt wird, so nähert sich dem Reed-Kontakt ein kleiner Permanentmagnet und somit schließt der Kontakt (bei der hier gezeigten Bauart).

Reed-Taster sind sehr zuverlässig, da sie nur aus sehr wenigen mechanischen Teilen aufgebaut sind. Die empfindlichen Kontaktstellen sind auch der Umwelt entzogen, da sie in einem hermetisch dichten Glasgehäuse untergebracht sind. Reed-Tasten sind aber nicht prellfrei. Wenn man einen solchen Taster betätigt, so entstehen im Moment des Schaltens viele kurze Impulse, die sich auf nachfolgende Schaltungen, die der Taster ansteuert, störend auswirken können.

Hg-Tasten

Bei diesen Tasten sind die Kontakte mit einem dünnen Quecksilberfilm benetzt. Dadurch wird ein prellfreies Schalten erreicht.

Kontaktlose Taster

Bei dieser Art von Tastern werden z. B. sogenannte Hallgeneratoren verwendet. Im Prinzip entspricht der mechanische Aufbau dem Reed-Taster, nur daß anstatt des Reed-Kontaktes der Hallgenerator in geeigneter Weise eingebaut wird.

Wenn der Taster betätigt wird, so verändert der Permanentmagnet die Lage zum Hallgenerator und es entsteht eine Hallspannung. Diese

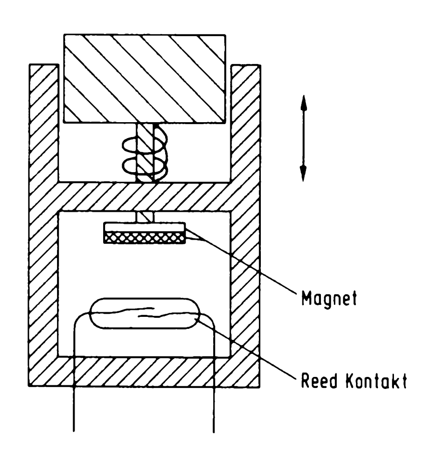


Abb. 6.1.1-1 Schematischer Aufbau eines Reed-Tasters

kann nun verstärkt und weiterverarbeitet werden. Der Vorteil dieser Tastenart liegt in seinem prellfreien Schalten und in seiner hohen Zuverlässigkeit. *Abb. 6.1.1-2* zeigt den schematischen Aufbau einer solchen Taste. Die Firma Rafi z. B. baut mit dem Hallgenerator gleich ein IC ein, das die Verstärkung übernimmt. Die Taste liefert dann beim Betätigen als Ausgangssignal einen LOW-Impuls. Derartige Tasten sind in der beim Datensichtgerät angegebenen alphanumerischen Tastatur und im kleinen Tastenfeld verwendet.

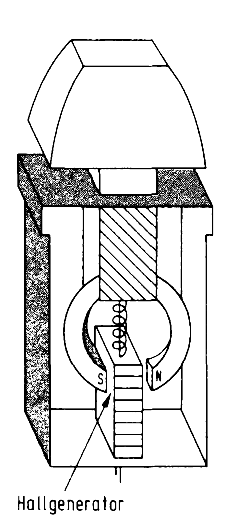


Abb. 6.1.1-2 Aufbau eines kontaktlosen Tasters

6.1.2 Entprellverfahren für einzelne elektromechanische Tasten

Entprellung mit RC-Glied

Eine solche Entprellschaltung zeigt *Abb. 6.1.2-1*. Die Schaltung ist mit einem handelsüblichen Schmitt-Trigger (SN 7413), einem Kondensator und einem Widerstand aufgebaut.

Im Ruhezustand, also wenn die Taste nicht betätigt ist, liegt am Eingang des Schmitt-Triggers ein LOW-Signal. Wenn der Taster betätigt wird, so wird der Kondensator schlagartig entladen und am Eingang des Schmitt-Triggers liegt ein HIGH-Signal. Im Moment des Betätigens springt dabei das Ausgangssignal von HIGH auf LOW (invertierende Funktion des verwendeten Schmitt-Triggers beachten). Wenn der Taster nun beim Betätigen prellt, so werden diese Impulse durch den Kondensator aufgefangen. Er wurde schon nach dem ersten Impuls entladen und kann sich innerhalb der kurzen Zeit, in der der Taster keinen Kontakt mehr gibt, aufgrund seines Prellens, nicht mehr auf eine genügend große Spannung aufladen. Dadurch wird die Schaltschwelle des Schmitt-Triggers nicht wieder erreicht.

Beim Öffnen des Schalters kann der Kondensator sich nach einer Weile wieder aufladen, so daß am Eingang des Schmitt-Triggers wieder ein LOW-Signal liegt. Damit wird wieder die Schaltschwelle des Schmitt-Triggers durchlaufen und am Ausgang des Schmitt-Triggers erscheint dann wieder ein HIGH-Signal.

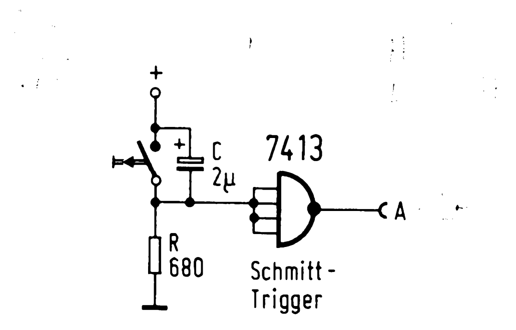


Abb. 6.1.2-1 Entprellschaltung mit RC-Glied

Entprellung durch ein RS-Flipflop

Diese Entprellschaltung zeigt *Abb. 6.1.2-2*. Man benötigt dazu einen Umtaster und zwei NAND-Verknüpfungen (1/2 SN 7400).

Im Ruhezustand erhält N 1 vom Taster ein LOW-Signal an den Eingang E 1. Am Ausgang der NAND-Verknüpfung tritt nur dann ein LOW-Signal auf, wenn am Eingang E 1 und gleichzeitig am Eingang E 2 ein HIGH-Signal auftritt.

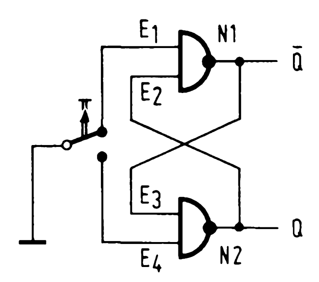


Abb. 6.1.2-2 Entprellung mit einem RS-Flipflop

Dieser Fall ist für N 1 bei nicht betätigtem Taster nicht erfüllt, am Ausgang Q von N 1 tritt also ein HIGH-Signal auf. Dieser Ausgang führt an den Eingang E 3 von N 2. Um auch hier den Ausgangspegel zu bestimmen, muß noch E 4 betrachtet werden. E 4 ist aber nicht beschaltet, da der Taster nicht geschlossen ist. Ein offener Eingang ist aber bei der TTL-Logik gleichwertig mit einem HIGH-Pegel an diesem Eingang. Bei N 2 ist also die NAND-Bedingung erfüllt. Am Ausgang Q liegt dann ein LOW-Pegel. Dieses Signal wird dann auch noch an E 2 von N 1 geführt.

Wenn der Taster gerade betätigt wird, so tritt zunächst der Fall ein, daß keiner der beiden Kontakte geschlossen ist. Das bedeutet an E 1 und an E 4 liegt nun ein HIGH-Pegel. (Genauer, es liegt scheinbar ein HIGH-Pegel an diesen Eingängen.) An E 2 liegt aber immer noch ein LOW-Pegel, so daß der Schaltzustand von N 1 unverändert bleibt. Auch N 2 bleibt unverändert.

Wird der Taster nun endgültig niedergedrückt, so erhält der Eingang E 4 ein LOW-Pegel und es ergeben sich neue Zustände. Der Ausgang von N 2 geht nun auf HIGH und nun erhält auch der Eingang E 2 ein HIGH-Pegel, so daß der Ausgang von N 1 auf LOW-Pegel schaltet. Der gleiche Vorgang geschieht umgekehrt, wenn der Taster wieder losgelassen wird. Wichtig dabei ist, zu erkennen, daß der Taster durch dieses bistabile Verhalten der Entprellschaltung entprellt wird.

6.2 Zehner-Tastatur

6.2.1 Mehrfach-Entprellung

Es wäre sehr umständlich, für jede Taste der Zehnertastatur eine eigene Entprellschaltung aufzubauen. Deshalb verwendet man für alle Tasten eine gemeinsame Entprellschaltung.

Abb. 6.2.1-1 zeigt ein Beispiel einer solchen Mehrfachentprellschaltung.

Dabei wird vor der eigentlichen Entprellung erst eine Umcodierung in den BCD-Code vorgenommen. Dies geschieht mit Hilfe einer Diodenmatrix. An den Leitungen TL 1 bis TL 4 stehen nun die codierten, aber noch nicht entprellten Signale. Bei Betätigung irgendeiner Taste der Zehner-Tastatur wird zusätzlich der Leitung TLK ein LOW-Signal zugeführt. Das Signal auf dieser Leitung, das ebenfalls nicht entprellt ist, wird nun einer aus T 1, C 1 und R 2 bestehenden Entprellschaltung zugeleitet. Das Impulsdiagramm in *Abb. 6.2.1-2* zeigt den prinzipiellen Ablauf einer vollständigen Entprellung. An M 2 liegt nun ein entprelltes, aber für die direkte Ansteuerung von TTL-Logik wegen der Anstiegszeit ungeeignetes Signal. Das Signal wird deshalb einem Schmitt-Trigger des Typs SN 7413 zugeführt. An M 3 liegt nun das TTL-kompatible entprellte Signal. Dieses Signal ist aber noch nicht geeignet, die Datenleitungen TL 1...TL 4 mit Hilfe der NAND-Verknüpfungen N 1...N 4 zu entprellen. Wenn das Signal von M 3 nämlich von HIGH auf LOW fällt, können noch Prellsignale auf den Leitungen TL 1...TL 4 sein.

Am besten ist es, die NAND-Verknüpfungen genau in der Hälfte der Betätigungszeit für eine kurze Weile freizugeben oder doch wenigstens dann, wenn sichergestellt ist, daß das Einschaltprellen aufgehört hat. Dies geschieht mit Hilfe der beiden Monoflops MV 1 und MV 2 des Typs SN 74121. Die abfallende Flanke des Signals an M 3 triggert dabei MV 1. Nach einer Zeit „t MV 1" kippt das Monoflop wieder zurück und triggert dabei MV 2. Das Ausgangssignal dieses Monoflops gibt nun

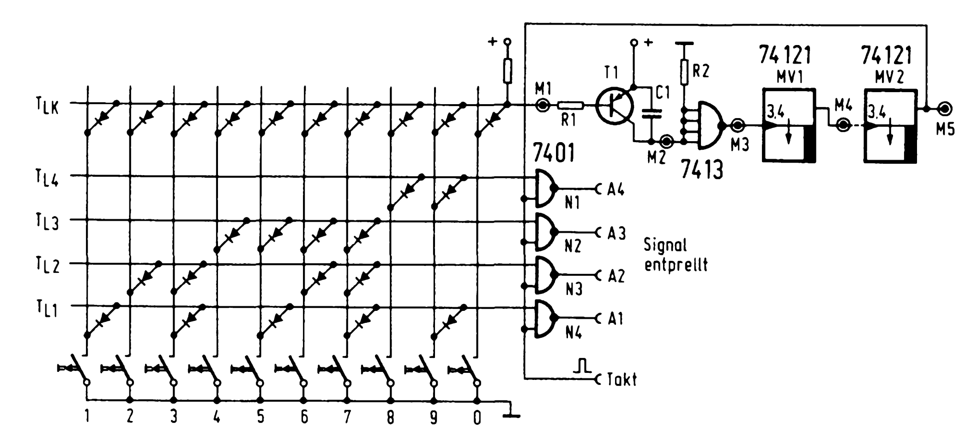


Abb. 6.2.1-1 Entprellen einer Zehnertastatur

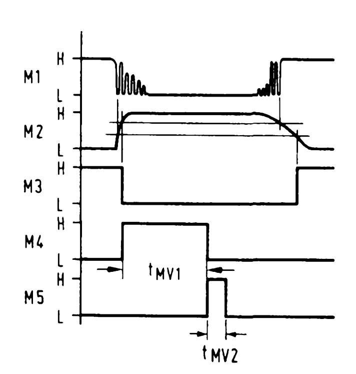


Abb. 6.2.1-2 Impulsdiagramm

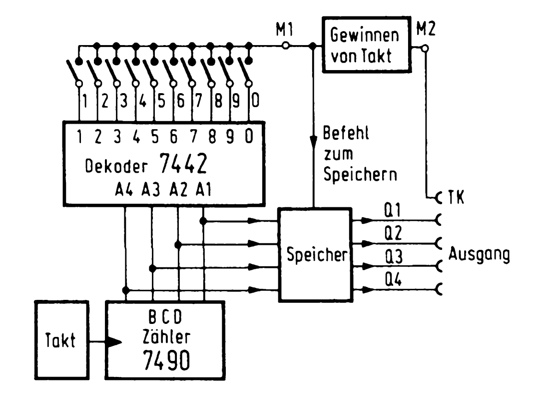


Abb. 6.2.2-1 Entprellen mit Hilfe des Abtastverfahrens

während der Zeit „t MV 2“ die NAND-Verknüpfungen frei.

Damit wurden die Signale auf den Leitungen TL 1...TL 4 auf den Ausgang der Schaltung gegeben.

6.2.2 Codierung mit Abtastverfahren

Außer der Codierung mit einer Diodenmatrix oder einem ROM in integrierter Technik gibt es auch noch ein anderes elegantes Verfahren zur Codierung von Tastaturen. Ein Prinzipschaltbild eines solchen Verfahrens zeigt *Abb. 6.2.2-1*. Die Tasten werden mit Hilfe eines Decodierers auf ihren Zustand (betätigt oder nicht betätigt) abgefragt. Der Decodierer wird von einem Zähler gesteuert, der im BCD-Code von 0 auf 9 zählt. Wird nun eine Taste, z. B. die Taste 5 betätigt, so erscheint nach einer kurzen Zeit an M 1 ein Impuls. Dies geschieht genau dann, wenn der Zähler an seinen Ausgängen den BCD-Code für die betätigte Taste hat, in dem Beispiel also den Code 0101 für die Taste 5.

Dieses Signal M 1 gelangt an einen Zwischenspeicher und gibt dort den Befehl, die Ausgangssignale des Zählers darin festzuhalten. Am Ausgang des Speichers steht also der Code der betätigten Taste fest an. Zur Weiter-

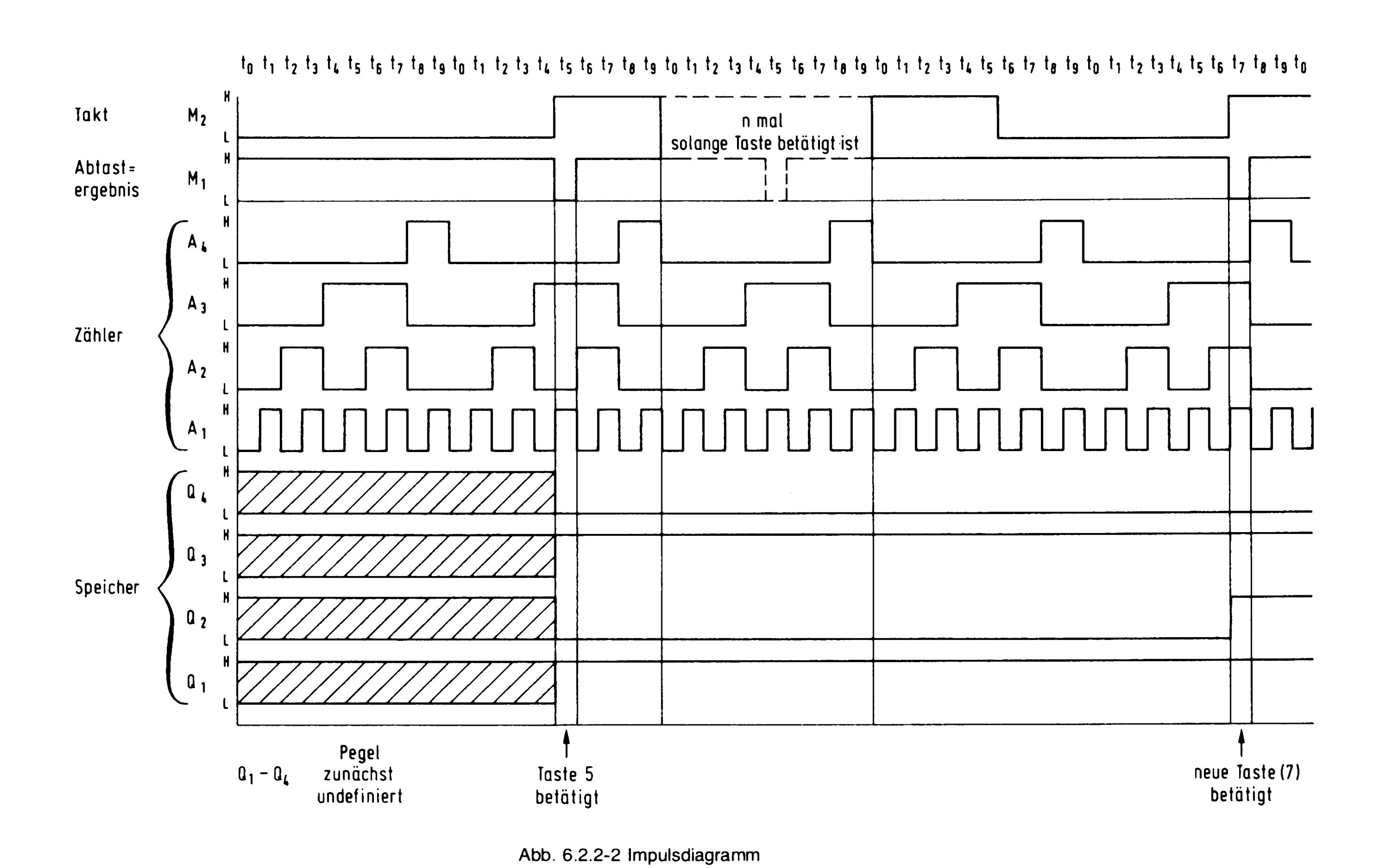


Abb. 6.2.2-2 Impulsdiagramm

verarbeitung dieser Information am Ausgang des Speichers wird nun noch ein Übernahmeimpuls benötigt, der kurz nach Betätigen des Tasters einmal entsteht. An der Leitung M 1 steht aber bei einer hohen Abfragefrequenz eine Impulsfolge bei Betätigen einer Taste an. Diese Impulsfolge kann aber relativ einfach dazu verwendet werden, einen geeigneten Taktimpuls zu erzeugen. Dazu kann dieses Signal an ein retriggerbares Monoflop (z. B. SN 74123) geführt werden, das eine entsprechende Zeitkonstante besitzt, und damit zu einem Ausgangssignal kommen.

Abb. 6.2.2-2 zeigt ein Impulsdiagramm für den Fall, daß Taste 5 betätigt wird und dann später Taste 7. Vor Betätigen der Taste 5 sind die Signalpegel der Ausgänge unbekannt bzw. nicht definiert. Dies ist in dem Impulsdiagramm durch die schraffierten Flächen bei den Ausgängen Q 1...Q 4 angedeutet. Meßpunkt M 2 gibt den Verlauf des Ausgangstaktimpulses an.

6.3 Alphanumerische Tastatur

6.3.1 Entprellung und Codierung mit LSI-Schaltkreisen

Um eine alphanumerische Tastatur zu codieren, wäre es sehr umständlich, eine diskrete Diodenmatrix aufzubauen. Die Industrie liefert komplette Tastaturcodier-ICs in LSI-Technik. Diese Tastaturcodier-ICs sorgen für das Entprellen der verwendeten Tastatur und natürlich für das Entstehen des Code eines gewünschten Codesystems.

Abb. 6.3.1-1 zeigt die Pin-Belegung eines solchen LSI-Schaltkreises. Der dargestellte Schaltkreis wird übrigens von General Instruments Corporation unter der Bezeichnung AY-5-3600 geliefert.

Das IC enthält einen ROM mit 3600 Speicherplätzen, die in 360 x 10 bit organisiert sind. Die Auswahl eines Wortes (10 bit) erfolgt durch die Betätigung einer der 90 möglichen Tasten. Ferner wird die Auswahl durch einen der vier Tastaturzustände bestimmt. Diese vier Tastaturzustände können „normal mode", „shift mode", „control" und „shift control" sein. Die Umschaltung der Zustände erfolgt mit Hilfe zweier Tasten, die mit auf der Tastatur angeordnet werden müssen. Ein Taster dient der Umschaltung von „normal mode" in „shift mode", ein anderer Taster der Umschaltung von „normal mode" in „control mode". Bei Betätigung beider Tasten erfolgt die Umschaltung in „shift control mode".

Die Taste „shift" dient, ähnlich wie bei der Schreibmaschine, hauptsächlich der Umschaltung von Klein auf Großschreibung sowie der Umschaltung mancher Sonderzeichen. Mit der Taste „control" kann man zusätzliche Steuercodes wie CR, LF... bestimmen. Die genaue Belegung der Tasten in den vier verschiedenen Zuständen für den ISO-7-bit-Code (Standard-Version des ICs AY-5-3600) zeigt *Abb. 6.3.1-2*.

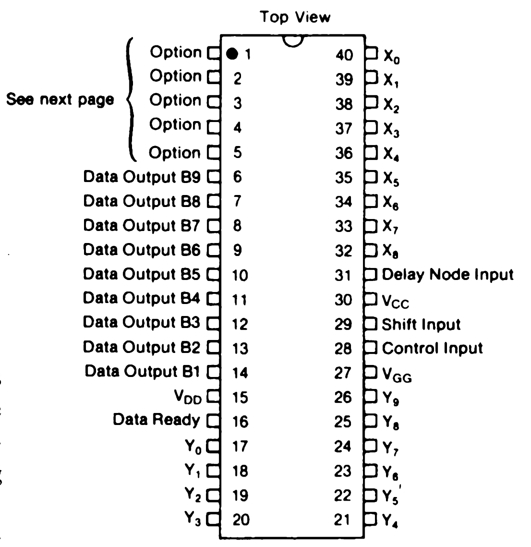


Abb. 6.2.1-1 Anschlußschema eines Tastaturcodier-ICs (AY-5-3600)

Abb. 6.3.1-2 Tastenbelegung

	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Y_0	1 < 1 SUB	2 @ 2 ETB	3 # 3 NAK	4 $ 4 DC4	5 % 5 STX	6 > 6 SOH	7 & 7 ETX	8 * 8 ESC	9 (9 EM
Y_1	q Q q DLE	w W w \	e E e DC3	r R r ENQ	t T t EOT	y Y y DC1	u U u BEL	i I i ACK	o O o |
Y_2	a A a @	s S s A	d D d B	f F f C	g G g D	h H h E	j J j F	k K k G	l L l X
Y_3	z Z z P	x X x Q	c C c R	SP SP SP SP	v V v S	b B b T	n N n U	m M m V	, , , ,
Y_4	HT HT HT I	RS RS RS FS	– – – ∧	CAN (CAN BS	ETX ETX ETX ETX	: * : SYN	= = = ~	/ ? / ,	. . . .
Y_5	H H H H	% % % %	$ $ $ $	CR CR CR M	} } } N	> > > Z	< < < W	' " ' "	; : ; :
Y_6	+ + + +	m | CR CR	L L L L	} } } K	? ? ? |	; + ; Y	p P p J	LF LF LF GS	| | | |
Y_7	SO > SO SO	SI SI SI SI	US US US US	VT VT VT VT	– = – –	NUL NUL NUL NUL	Ø) Ø DC2	= + = +	– – – –
Y_8	P @ NUL NUL	n ∧ SO SO	6 6 ACK ACK	7 ' BEL BEL	))))	* * * *	& & & &	FF < FF FF	Ø Ø Ø Ø
Y_9	1 ! SOH SOH	2 " STX STX	k | DEL DEL	" " " "	SP SP SP SP	! ! ! !	# # # #	((((	9) HT HT

1	2
3	4

1 = normal 2 = shift 3 = control 4 = shift control

Abb. 6.3.1-3 Schaltbild einer Tastatur mit dem IC AY-5-3600

Abb. 6.3.1-3 zeigt übrigens noch ein Anschlußschema und das „Innenleben" des ICs.

Das IC besitzt 10 Datenausgänge. Da der ISO-7-bit-Code nur 7 Datenleitungen vorsieht, ist es notwendig, aus diesen 10 Datenleitungen die richtigen auszuwählen. *Abb. 6.3.1-4* zeigt die Zuordnung der Datenleitungen zu den Datenleitungen des ISO-7-bit-Codes. An den nicht angeschlossenen Datenleitungen des ICs liegen zusätzliche Informationen, die aber für den ISO-7-bit-Code nicht relevant sind.

6.3.2 Rafi-Tastatur (mit prellfreien Tasten)

Eine Tastaturcodierung, wie sie von der Firma Rafi als Standard-Codierung verwendet wird, zeigt *Abb. 6.3.2-1*. Die Tasten selbst, die in diesem Bild nicht eingezeichnet sind, werden ähnlich wie in dem vorhergehenden Abschnitt über eine Matrix verknüpft, mit dem Unterschied, daß im Falle der prellfreien Tasten von Rafi, die zwei Aktiv-LOW-Ausgänge besitzen, eine umfangreiche Abtastschaltung entfällt. Die Signale für Reihe und Zeile können dann direkt codiert werden, dadurch ist es

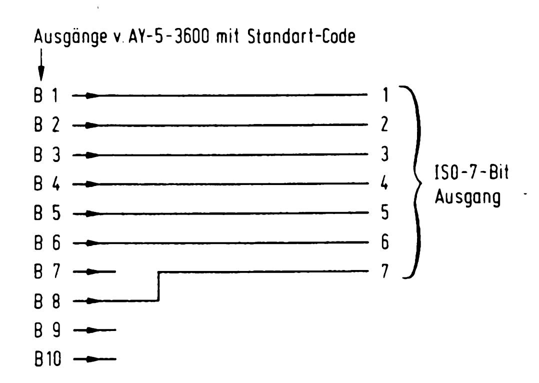
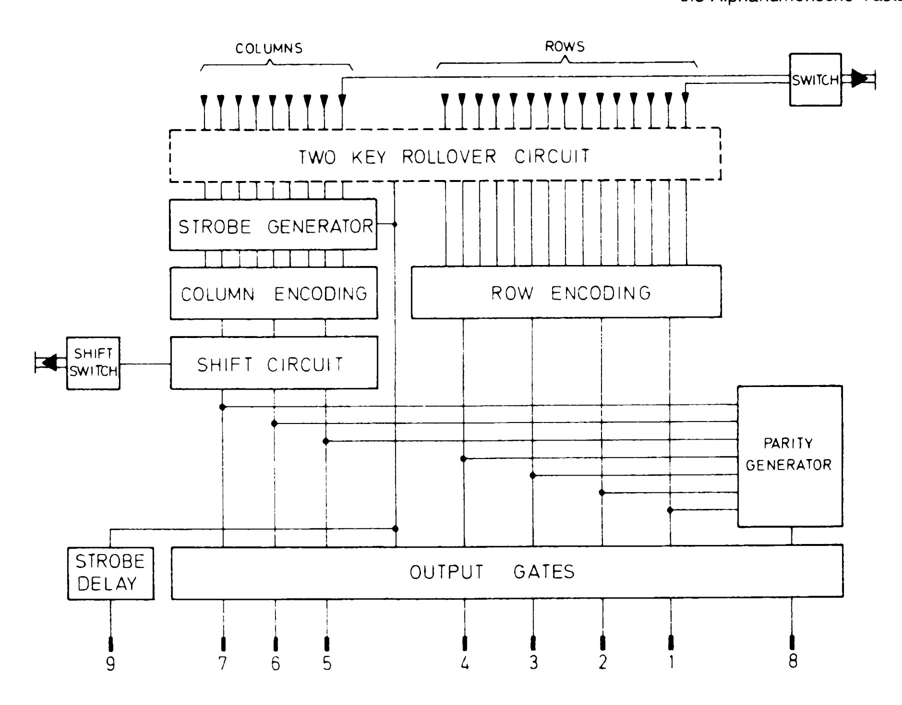


Abb. 6.3.1-4 Gewinnen der Datenleitungen für den ISO-7-Bit-Code



Technische Daten der Standard-Codierung:

Betriebsspannung:	5 V ± 5%
Stromverbrauch:	je Taste 6 mA, Codierung ca. 200 mA, 64-teilige Tastatur ca. 800 mA
Ausgänge:	negative Logik, TTL / DTL kompatibel log. „0" ≧ 2,4 V / 0,4 mA, log. „1" ≦ 0,4 V / 16 mA
Anstiegs- und Abfallzeit:	≦ 100 ns
Informationsausgänge:	7 Bit und 1 Paritätsbit (gerade oder ungerade), Datenausgabe parallel
Codierung:	ECMA-Standard (US ASC II Code, DIN 66 003), Mono, Dual- und Tri-Funktion
Elektrische Verriegelung:	2 key oder N-key roll-over (nur codierte Tasten)
Strobe-Ausgang:	ca. 50 ns verzögert, dynamische Impulsdauer 100 μs
Funktionsausgänge:	Shift, Shift Lock, CTRL, weitere Tasten auf Wunsch
Anwendungsklasse:	KTF (DIN 40 040)
Arbeitstemperatur:	0 . . . + 65° C
Lagertemperatur:	− 40 . . . + 70° C

Abb. 6.3.2-1 Schematik einer Codierung von Rafi

möglich, auf einen LSI-Schaltkreis unter Umständen zu verzichten.

Abb. 6.3.2-2 zeigt zwei Anordnungen der Tasten auf einer Tastatur, wie sie von Rafi geliefert wird.

a) zeigt eine Tastatur, wie sie beim Bau des Datensichtgerätes angegeben wurde, sie besitzt keine CTRL-Taste (Control-Taste).

b) zeigt eine komfortablere Tastatur mit Control-Taste und Sondertasten.

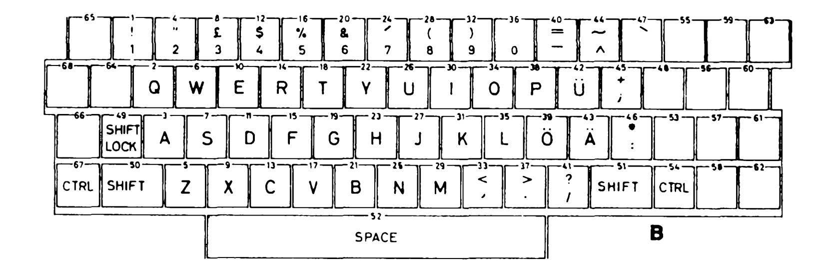
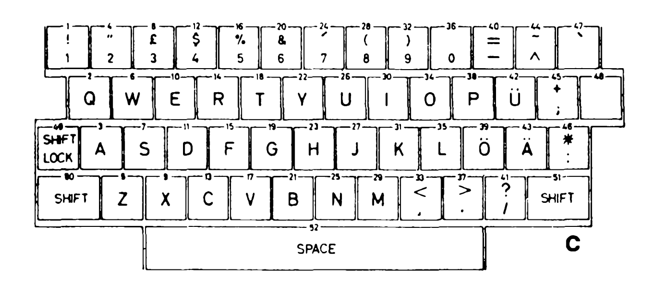


Abb. 6.3.2-2a–b Verschiedene Tastenanordnungen (Rafi)



6.4 Teletype

Es soll hier noch kurz auf ein industrielles Ein- und Ausgabegerät eingegangen werden, die Teletype (Fernschreiber für Computeranwendung). *Abb. 6.4-1* zeigt das Bild einer solchen Teletype (Typ ASR 33, automatic send receive).

Sie gestattet die Verarbeitung von 10 Zeichen in der Sekunde. Als Codierung wird der ISO-7-bit-Code verwendet. Mit einer solchen Teletype (mit dem hier gezeigten Modell) ist es auch möglich, einen Lochstreifen zu erstellen, und es ist auch möglich, einen solchen Lochstreifen wieder einzulesen.

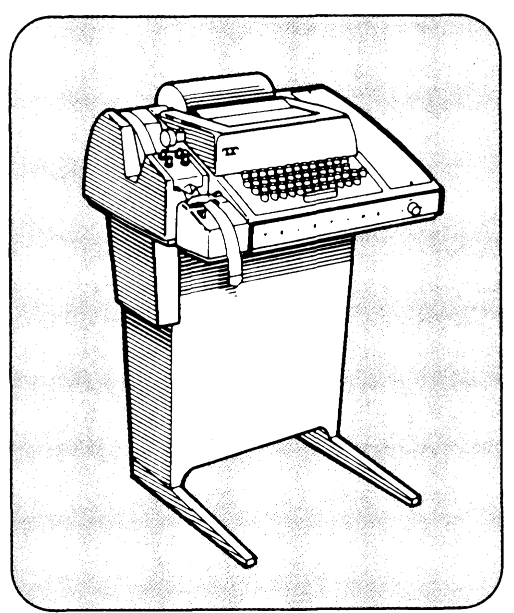




Abb. 6.4-1 Teletype Typ ASR 33

7 Periphere Speicher

7.1 Lochstreifen als Speichermedium

Relativ häufig tritt auch noch der Lochstreifen als Speichermedium auf. In der Computertechnik verwendet man heute ausschließlich den 8-bit-Lochstreifen. Es ist nämlich möglich, in einer Spalte genau 1 Byte Information unterzubringen.

Abb. 7.1-1 zeigt den schematischen Verlauf eines Speichervorgangs auf Lochstreifen und die Anordnung der Daten auf dem Lochstreifen, wie sie zum Beispiel bei Ausgabe des ISO-7-bit-Codes verwendet wird.

Nachteilig beim Lochstreifen ist die große Zugriffszeit, man verwendet ihn also nur noch zum Speichern von Daten und Programmen. In der Anfangszeit der Computertechnik wurde er als aktiver Speicher für das Programm verwendet. Vorteil ist das recht störsichere Arbeitsprinzip.

7.2 Interface für einen Bandspeicher mit Kassettenrekorder

7.2.1 Verschiedene Aufzeichnungsverfahren

Hier sollen zwei industrielle Aufzeichnungsverfahren erklärt werden, die repräsentativ für zwei verschiedene Aufzeichnungskategorien sind.

a) Zweispuraufzeichnung

Bei diesem Verfahren wird auf einer Spur die eigentliche Information aufgezeichnet und auf der zweiten Spur wird der Takt mitgeführt. *Abb. 7.2.1-1* zeigt ein Beispiel für ein nach

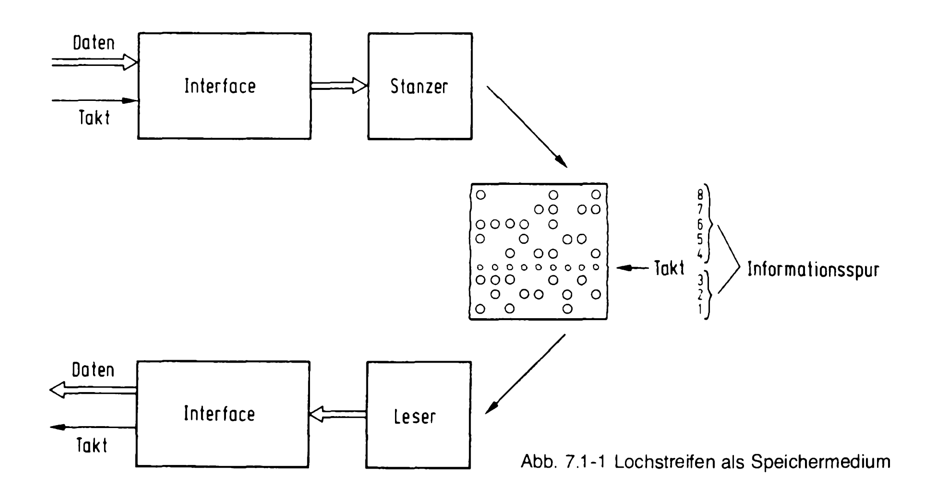


Abb. 7.1-1 Lochstreifen als Speichermedium

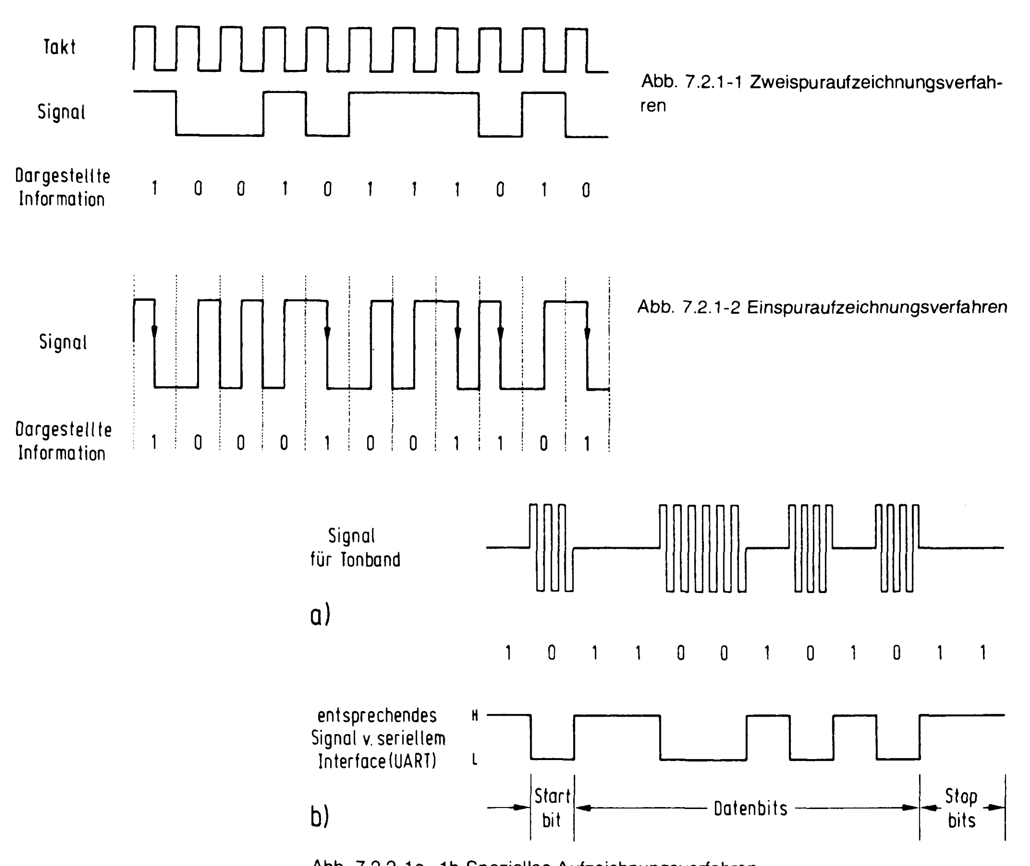


Abb. 7.2.1-1 Zweispuraufzeichnungsverfahren

Abb. 7.2.1-2 Einspuraufzeichnungsverfahren

Abb. 7.2.2-1a, -1b Spezielles Aufzeichnungsverfahren

dem NRZ-(C)-(non return to zero change [Richtungsschrift])Verfahren gebildetes Signal, das der einen Spur des betreffenden Bandspeichers zugeführt wird. Der anderen Spur wird das Taktsignal zugeführt.

b) Einspuraufzeichnung
Etwas komplizierter ist es, eine Einspuraufzeichnung vorzunehmen, da in der gleichen Spur sowohl der Takt, als auch die Information aufgezeichnet werden muß.

Abb. 7.2.1-2 zeigt ein Beispiel für ein nach dem Bi-0-(bi phase level)Verfahren gebildetes Signal. Bei diesem Verfahren wird ein Signal am Ausgang invertiert, wenn ein Sprung der darzustellenden Information von 1 auf 0 oder von 0 auf 1 erfolgt.

7.2.2 Einfache Verwirklichung eines Bandspeichers mit störsicherem Verfahren

Die vorher gezeigten Verfahren sind entweder zu aufwendig für die Realisation oder sie erlauben den Einsatz eines Kassettenrekorders nicht in bezug auf Anforderung der Qualität der Aufzeichnung und würden dann vielleicht nur eine sehr geringe Aufzeichnungsdichte erlauben. Hier soll nun ein Verfahren erklärt werden, das eine recht hohe Aufzeichnungsdichte erlaubt und außerdem ziemlich störsicher ist.

Abb. 7.2.2-1 zeigt das Ausgangssignal, das dem Kassettenrekorder direkt zugeführt werden kann. Es handelt sich dabei um ein Einspuraufzeichnungsverfahren. Ferner ist das

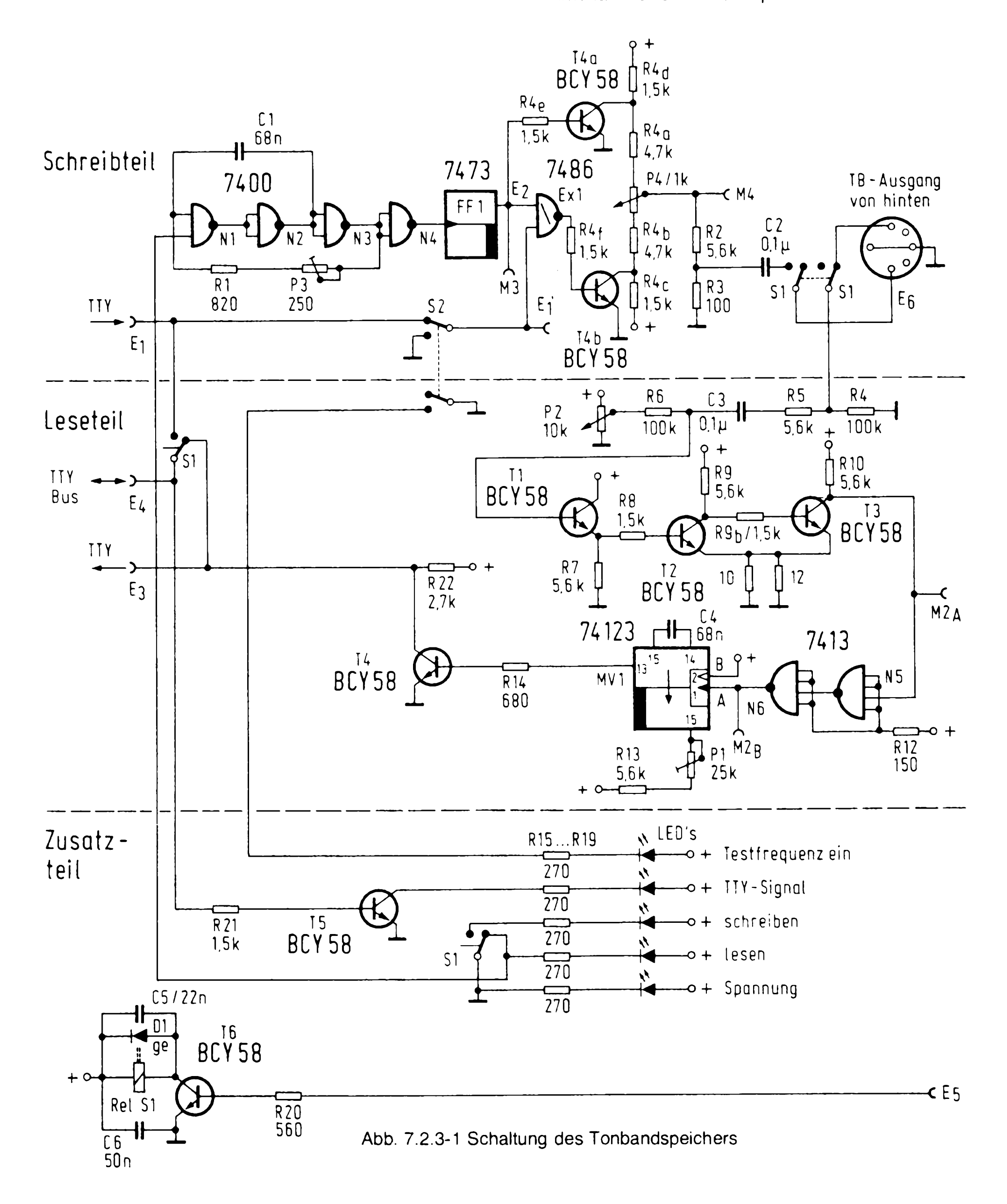


Abb. 7.2.3-1 Schaltung des Tonbandspeichers

Ausgangssignal gleichspannungsfrei. Dies ist eine wichtige Voraussetzung für eine störsichere Aufzeichnung mit einer hohen Speicherdichte. Die eingehende Information stammt direkt von einem Ausgang eines asynchron arbeitenden seriellen Interface (siehe Kapitel Datensichtgerät). Damit fällt eine spezielle Schaltung zur Aufbereitung eines Taktsignals weg. Das Signal kann bei der Wiedergabe von dem Tonbandgerät auf einfache Art und Weise so aufbereitet werden, daß es einem seriellen Interface wieder direkt zugeleitet werden kann.

7.2.3 Aufbau der Aufnahmeschaltung (Schreiben)

Abb. 7.2.3-1 zeigt die gesamte Schaltung des Tonbandspeichers. Die Aufnahmeschaltung ist im Bild oben durch eine dünne Linie abgegrenzt erkennbar. Links oben erkennt man den Taktgenerator, bestehend aus N 1...N 4, C 1, R 1 und P 3. Das Taktsignal wird einem Flipflop FF 1 des Typs SN 7473 zugeführt, das die Aufgabe hat, aus dem Taktsignal eine symmetrische Taktfrequenz mit halber Frequenz zu erzeugen. Am Meßpunkt M 3 kann diese Taktfrequenz abgegriffen und mit dem Trimmer P 3 auf 4 kHz abgeglichen werden. Die Frequenz des Oszillators beträgt dann 8 kHz.

Die Trägerfrequenz von 4 kHz wird dann einem speziellen Modulator zugeführt, der aus den Bauelementen Ex1, T4a, T4b, R4a, R4b, R4c, R4d, R4e, R4f und P 4 besteht.

Das Signal des seriellen Interface wird dem anderen Eingang von Ex 1 zugeführt.

Es wird angenommen, dieser Eingang liegt auf einem HIGH-Pegel. Dann erscheint das Trägersignal, das ja dem anderen Eingang von Ex 1 zugeführt wird, an seinem Ausgang aufgrund der Wahrheitstafel der Verknüpfung Ex 1 invertiert. Dieses Signal wird nun noch einmal durch den Transistor T 4b invertiert und gelangt dann an den einen Anschluß von P 4. Außerdem gelangt das Trägersignal noch einmal durch T 4a invertiert an den Anschluß des Potentiometers (oder Trimmers) P 4. P 4 muß nun so abgeglichen sein, daß am Ausgang von P 4 (Meßpunkt M 4) kein Trägersignal mehr feststellbar, zumindest vernachlässigbar klein ist.

Wird angenommen, daß der eine Eingang der Verknüpfung Ex 1 an einem LOW-Pegel liegt, so erscheint dann die Trägerfrequenz am Ausgang von Ex 1 in ihrer ursprünglichen Form, nicht invertiert. An M 4 erscheint nun das Trägersignal. Dieses Modulationsergebnis wird nun noch einem Spannungsteiler zugeführt, der das Signal in seiner Amplitude stark abschwächen soll, so daß es dem Eingang eines Tonbandgerätes zugeführt werden kann (Radio-Eingang). Zuvor wird das Signal aber noch über einen Kondensator C 2 geführt, so daß die Gleichspannungskomponente beseitigt wird.

7.2.4 Aufbau der Wiedergabeschaltung (Lesen)

Der mittlere abgegrenzte Teil im Schaltbild zeigt die Wiedergabeschaltung.

Das Ausgangssignal des Tonbandgerätes gelangt über R 5 und C 3 an die Basis von T 1. Gleichzeitig gelangt von P 2 über R 6 ein Gleichspannungssignal an die Basis von T 1. Mit P 2 läßt sich der Triggerpegel einstellen. Die Stufe T 1 und R 7 stellt einen Impedanzwandler dar, dessen Ausgangssignal einer diskreten Schmitt-Trigger-Stufe zugeführt wird. Mit zwei zusätzlichen Betriebsspannungen ist auch ein OP (z. B. 741) verwendbar, der als Schmitt-Trigger geschaltet ist. Das Signal wird dann einer weiteren Schmitt-Trigger-Stufe zugeführt, die die Aufgabe hat, das Signal endgültig TTL-kompatibel zu machen (N 5 und N 6 des Typs SN 7413). Das Ausgangssignal dieser Stufe wird nun einem retriggerbaren Monoflop zugeführt, das die Aufgabe hat, die immer noch vorhandene Trägerfrequenz auszublenden. Ein Feinabgleich der Verweilzeit kann mit P 1 vorgenommen werden. Das Ausgangssignal gelangt anschließend an die Schaltstufe, die aus R 14 und T 4 besteht.

Abgleichmaßnahmen

1. Mit P 3 die Frequenz an M 3 auf 4 kHz abgleichen.
2. Mit P 4 das Signal an M 4 auf Wechselspannungsfreiheit kalibrieren, wenn der Eingang E 1' von Ex 1 auf HIGH-Pegel ist.
3. Aufnahme mit dem Tonbandgerät durchführen, dabei günstigen Aussteuerungswert experimentell ermitteln.

4. Mit P 2 den Pegel am Meßpunkt 2a so einstellen, daß im Ruhezustand, das heißt bei nicht vorhandenem Eingangssignal der Pegel an diesem Meßpunkt auf HIGH ist.

5. Mit P 1 die Verweilzeit von MV 1 so einstellen, daß bei vorhandenem Träger das Signal an M 1 sicher auf HIGH schaltet und keine Impulse mehr vorhanden sind, die von dem Träger herrühren. Mit S 2 ist es möglich, schon bei der Aufnahme ein Testsignal in Form eines Dauerträgersignals auf das Tonband aufzuzeichnen, so daß der beschriebene Abgleich mit P 1 leicht möglich ist.

Beim Abgleich verfährt man so, daß bei Einstellung von P 1 zunächst die Stellung gesucht wird, bei der die Trägerfrequenz in Form von Impulsen noch an M 1 erscheint. Dann wird der Trimmer so verstellt, daß diese Impulse gerade verschwinden.

Die eventuell notwendig werdenden genaueren Einstellungen der Trimmer können nur bei angeschlossenem seriellen Interface vorgenommen werden. Das Aufzeichnungsverfahren erlaubt dabei Übertragungsgeschwindigkeiten bis zu 1200 Baud. Dabei wird ohne Paritätsbit mit zwei Stopp-Bits und mit 8 Datenbits gearbeitet (siehe auch die Beschreibung des seriellen Interfaces im Kapitel Datensichtgerät).

Die ausgeführte Schaltung in Abb. 7.2.3-1 enthält außer den beschriebenen Schaltungseinzelheiten auch noch solche, die den Bedienungskomfort erhöhen. Mit S 3 ist es möglich, ein Relais zu betätigen, das die Schaltung von dem Aufnahme- in den Wiedergabezustand versetzt. Durch die Verwendung des Relais ist es möglich, die Schaltung auch fernzubedienen. Dies geschieht über den Eingang E 5. Ein an diesen Eingang gelegtes HIGH-Signal schaltet das Relais ein, und damit die Schaltung des Tonbandspeichers von dem Wiedergabe- in den Aufnahmezustand. Die LEDs zeigen den jeweiligen Betriebszustand der Schaltung an. Die Schaltung besitzt noch einen gemeinsamen Bus-Ein-/Ausgang. Er ist mit E 4 bezeichnet. Wenn mit dem Mikrocomputersystem und mit dem Datensichtgerät gearbeitet wird, so ist es möglich, diesen Bus-Ein-/Ausgang E 4 mit dem Eingang des Datensichtgerätes und mit dem Ausgang des Mikrocomputers zu verbinden. Dann wird eine Abspeicherung von Programmen ermöglicht (in Kapitel 9.8.2 bei der Beschreibung des Dienstprogramms 2 wird weiteres dazu erklärt).

Steht ein größeres Tonband zur Verfügung, das eine Aufnahme- und Wiedergabegeschwindigkeit von 19 cm/s zuläßt, so ist es möglich, die Trägerfrequenz zu erhöhen. Dazu wird für C 1 ein Kondensator mit dem Wert 33 nF gesetzt. Die Trägerfrequenz wird dann an Meßpunkt M 3 auf 8 kHz abgeglichen. Durch diese Maßnahme ergibt sich eine viel größere Störsicherheit als bei Verwendung eines normalen Kassettenrekorders.

8 Der Mikrocomputer

Früher waren Computersysteme groß und teuer. Heute ist es möglich geworden, einen in der Wirkungsweise dem großen Computer ähnlichen Computer, den Mikrocomputer in moderner LSI-Technik auf einem Chip zu verwirklichen. Mit Hilfe des Mikrocomputers können Digitalschaltungen verwirklicht werden, die bisher einen hohen schaltungstechnischen Aufwand bedeutet haben.

Vor der „Erfindung" des Mikrocomputers konnten, um hohe Packungsdichten in einem Gerät zu erreichen, kundenspezifische LSI-Schaltungen angefertigt werden. Die Anfertigung solcher LSI-Schaltkreise war aber nur bei hohen Stückzahlen lohnend. Für Einzelgeräte, die gefertigt werden sollten, war diese Methode nicht rentabel. Einzelgeräte mußten also in den meisten Fällen mit Schaltkreisen der SSI- und MSI-Technik gefertigt werden. Der Mikrocomputer brachte die große Wende. Er kann in großen Stückzahlen hergestellt werden, und er ermöglicht eine individuelle und flexible Verwendung durch den Benutzer.

Durch Programme, die in handelsüblichen Festwertspeichern abgelegt werden können, und mit ein paar zusätzlichen Peripheriebausteinen läßt sich ein solches Mikrocomputersystem zum Beispiel in ein rechnendes Frequenzmeßgerät, in eine Registrierkassensteuerung, in eine Steuerung für ein intelligentes Terminal oder in eine Nähmaschinensteuerung verwandeln.

Der Phantasie sind bei der Erfindung von Anwendungsbeispielen fast keine Grenzen gesetzt. Nur die Verarbeitungsgeschwindigkeit setzt Grenzen. Bei Geräten, bei denen es auf eine hohe Verarbeitungsgeschwindigkeit ankommt, kann der Mikrocomputer noch nicht eingesetzt werden.

Im Handel gibt es die unterschiedlichsten Arten von Mikroprozessoren. Mit Mikroprozessor ist im Gegensatz zum Begriff Mikrocomputer nur das IC gemeint, das die CPU (Zentrale Steuerung) darstellt. Es sind also die zum vollständigen Mikrocomputer notwendigen Speicher und Peripherieeinheiten ausgenommen.

Es gibt Mikroprozessoren, die eine Wortlänge von 4 bit, 8 bit, 12 bit, 16 bit verarbeiten. Unter Wortlänge versteht man die Anzahl der Bits, die von einem Mikroprozessor zusammenhängend verarbeitet werden können.

Mikroprozessoren können ferner unterschiedliche Befehlssätze, unterschiedliche Technologien und unterschiedliche Arbeitsgeschwindigkeiten haben und gegebenenfalls mikroprogrammierbar sein. Bei einem Mikroprozessor, der mikroprogrammierbar ist, kann durch ein entsprechendes Mikroprogramm eine Softwarekompatibilität zu einem anderen Mikrocomputer erreicht werden, falls der andere Mikrocomputer die gleiche Wortlänge besitzt, oder es kann durch ein entsprechendes Mikroprogramm eine besonders gute Anpassung an eine bestimmte Aufgabe erzielt werden. Gewählt wurde hier ein Mikroprozessor, der sich auch schon in der Industrie gut einge-

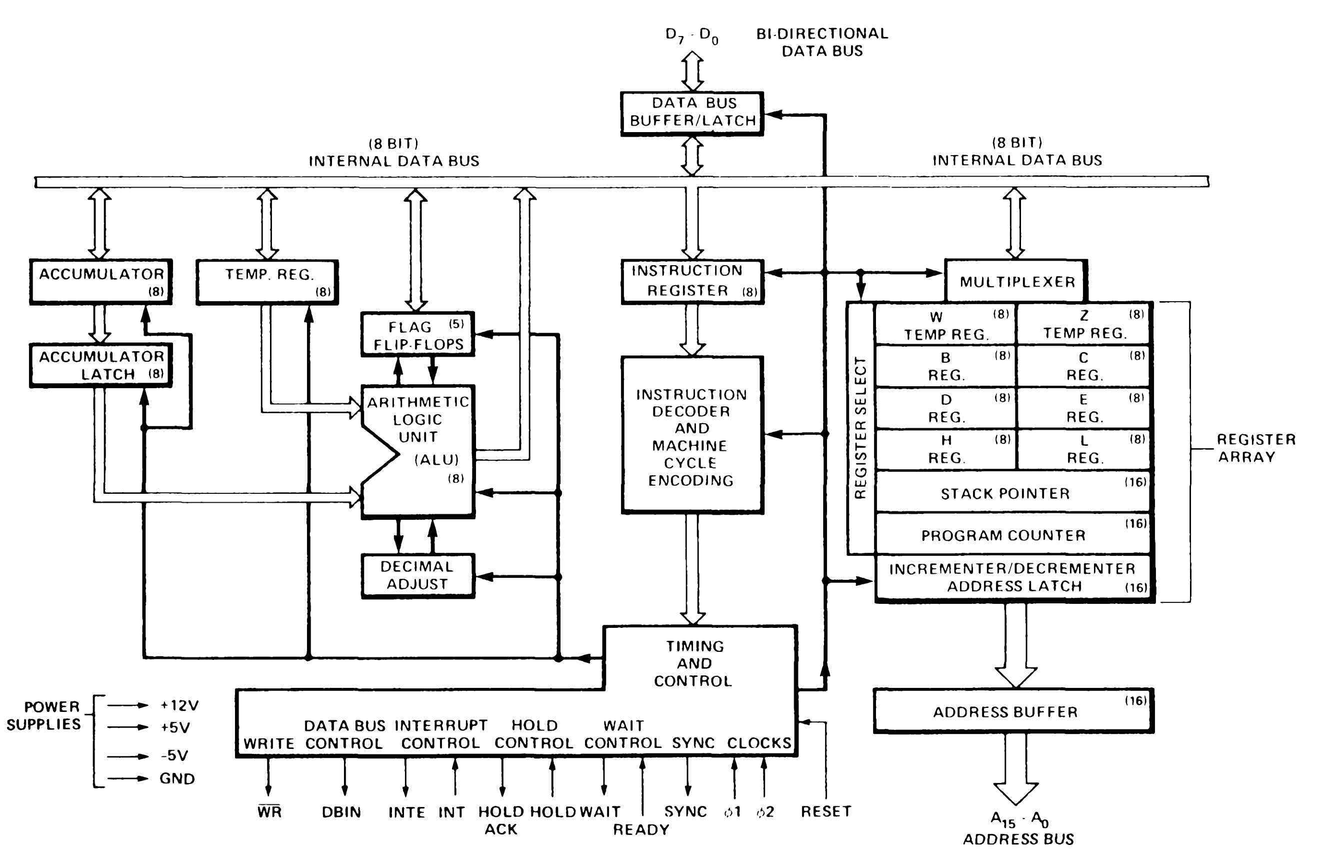


Abb. 8.1-1 Prinzipschaltbild des Mikroprozessors 8080

führt hat, der Mikroprozessor mit der Typenbezeichnung 8080. Es handelt sich hier um einen Mikroprozessor mit einer Wortlänge von 8 bit und mit der Fähigkeit, einen Speicher von 64 KBytes direkt und parallel zu adressieren. Ferner können 256 verschiedene Peripherieeinheiten adressiert werden. Der Mikroprozessor 8080 erlaubt eine recht hohe Verarbeitungsgeschwindigkeit und besitzt einen komfortablen Befehlssatz.

8.1 Wirkungsweise und Aufbau

Die 8080 CPU kann in vier verschiedene Funktionseinheiten eingeteilt werden. Eine Registermatrix und eine Adreßlogik, eine arithmetische Logik-Einheit, das Befehlsregister und die Steuerlogik sowie der Bi-Direktionale-Datenbus-Puffer sind diese Einheiten. Diese Funktionsblöcke sind in *Abb. 8.1-1* erkennbar.

Die CPU besitzt unterschiedliche Register. Es sind dies der Programmzähler (16 bit), das Stapelregister (auch Stack Pointer genannt) (16 bit), ferner 6 Register mit je 8 bit, die auch paarweise angesprochen werden können: Register B und C, Register D und E und Register H und L. Schließlich sind noch die Register W und Z, die als Zwischenspeicher dienen und dem Benutzer nicht zur Verfügung stehen, vorhanden. Die „Arithmetische Logik-Einheit" enthält ebenfalls verschiedene Register, von denen für den Benutzer nur der Akkumulator (8 bit) und ein Zustandsregister mit 5 bit direkt zugänglich sind.

Das Zustandsregister beinhaltet die Information über „Nullbit", „Übertragsbit", „Paritätsbit", „Vorzeichenbit" und „Hilfsübertragsbit" (siehe auch nächsten Abschnitt). Die Pinbelegung zeigt *Abb. 8.1-2*. Eine ausführliche Beschreibung findet man in dem entsprechenden User Manual von Intel (bzw. Siemens).

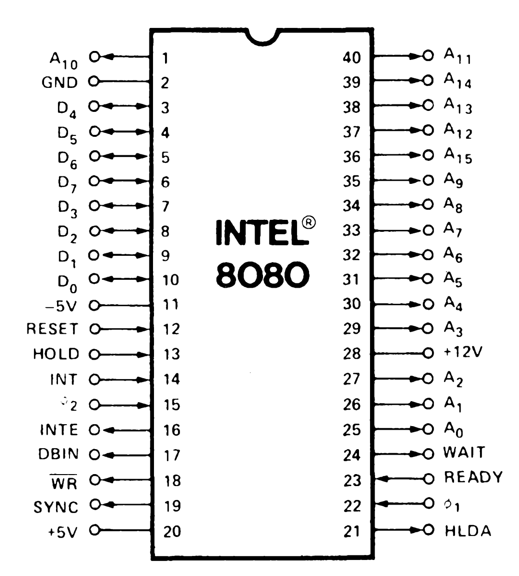


Abb. 8.1-2 Anschlußschema des ICs 8080

8.2 Die Befehlsstruktur des Mikroprozessors 8080

Wie schon gesagt, besitzt der Mikroprozessor 8080 ein 5-bit-Zustandsregister. Der Zustand dieses Registers gibt über bestimmte Situationen bei den Datenbits (z. B. Inhalt des Registers A) Auskunft. Die Bedeutung der einzelnen Bitstellen wird anschließend erläutert.

1. Das Übertragsbit (carry bit):
Das Übertragsbit wird bei manchen Datenoperationen gesetzt oder rückgesetzt.

2. Das Hilfs-Übertragsbit (auxiliary carry bit):
Diese Bitstelle wird bei einem Übertrag des vierten Bits eines Datenwortes verändert. Man verwendet dieses Bit bei Dezimaladditionen zur direkten Dezimalkorrektur.

3. Vorzeichenbit (sign bit):
Mit einem Byte können normalerweise Zahlen zwischen 0 und 256 dargestellt werden. Es ist aber auch noch eine andere Einteilung möglich. Dabei wird der Bereich −128...+127

überstrichen. Das höherwertigste Bit eines Bytes stellt dabei das Vorzeichenbit dar. Das Vorzeichenbit des Mikroprozessors wird gemäß dieser Definition bei manchen Operationen gesetzt bzw. rückgesetzt.

4. Das Nullbit (zero bit):
Dieses Bit wird bei der Ausführung von manchen Befehlen gesetzt, wenn das Ergebnis Null ist.

5. Das Paritätsbit (parity bit):
Die Parität wird bei manchen Operationen geprüft und das Paritätsbit entsprechend verändert. Das Bit wird gesetzt, wenn die Parität gerade ist und bei ungerader Parität rückgesetzt.

Die Überprüfung des Zustands aller Bits kann softwaremäßig, also durch ein Programm, erfolgen. Es folgt die Beschreibung des Befehlssatzes des Mikroprozessors 8080.

1. Übertragsbit Befehle (carry bit instructions)
CMC
Komplementiere das Übertragsbit (complement carry)
Code: 3FH
(H heißt Hexadezimal; diese Abkürzung wird in dem Mikroprozessor-Handbuch von Intel definiert.)
Wenn das Übertragsbit vor der Ausführung eines Befehls gesetzt ist (also 1), so wird es nach der Ausführung des Befehls rückgesetzt (auf 0) und umgekehrt. Andere Zustandsbits werden nicht verändert.

STC
Setze das Übertragsbit (set carry)
Code: 37H
Hier wird das Übertragsbit gesetzt (1). Außer dem Übertragsbit werden auch hier keine anderen Zustandsbits verändert.

2. Einzelregister-Befehle (single register instructions)
Die CPU besitzt sieben einzelne Register:
Register A:
Dieses Register stellt den Akkumulator der CPU dar. In ihm werden die arithmetischen Operationen ausgeführt.
Register B, C, D, E:
Sie können als Zwischenergebnisspeicher, als Zähler oder als Adreßregister verwendet werden. Es ist auch möglich, sie paarweise auf einmal zu adressieren (Registerpaar B und C, sowie Registerpaar D und E).
Register H, L:
Sie können ebenfalls wie die Register B...E verwendet werden, nur daß den Registern als Registerpaar noch eine besondere Bedeutung zufällt. Dabei dienen die Register als Adreßspeicher (H beinhaltet die höherwertige Stelle und L die niederwertige Stelle einer Adresse). Es können dann mit besonderen Befehlen Daten auf einfache Weise im Speicher abgelegt werden. Ferner besitzt die CPU noch zwei Registerpaare, die nicht einzeln angesprochen werden können.
Registerpaar SP:
Dieses Registerpaar wird als Adreßspeicher für den Stapelregisterbereich im Hauptspeicher verwendet. Dort werden Rücksprungadressen untergebracht, sowie die Registerinhalte der anderen Register (A, B, C, D, E, H, L und Zustandsbits). Dabei können immer zwei Register mit einem Befehl dorthin geschafft werden (also A und Zustandsbits, B und C...). Dazu wird später noch genaueres ausgeführt.
Registerpaar PC:
Dieses Register wird ausschließlich als Programmzähler verwendet.

INR...
Erhöhe den Inhalt des angegebenen Registers oder der Speicherzelle (mit durch den Inhalt des Registerpaars HL gegebenen Adresse) um eins (increment register or memory).
INR A Code: 3CH, INR B Code: 04H,
INR C Code: 0CH, INR D Code: 14H,
INR E Code: 1CH, INR H Code: 24H,
INR L Code: 2CH
Die Ausführung dieser Befehle erhöht den Inhalt des angegebenen Registers um eins.

INR M Code: 34

Die Ausführung dieses Befehls erhöht den Inhalt der Speicherzelle, deren Adresse durch den Inhalt des Registerpaars HL gegeben ist.

Beispiel: Enthält Register H den Wert 12H und Register L den Wert 2FH, so wird bei Ausführung des Befehls INR M der Inhalt der Speicherzelle mit der Adresse 122FH um eins erhöht. Enthielt die Speicherzelle beispielsweise den Wert E9H, so enthält sie dann den Wert EAH.

Bei Ausführung eines der INR-Befehle werden alle Zustandsbits außer dem Übertragsbit beeinflußt.

DCR...

Verringere den Inhalt des angegebenen Registers oder der Speicherzelle (mit durch den Inhalt des Registerpaars HL gegebenen Adresse) um eins.
(decrement register or memory)

DCR A Code: 3DH, DCR B Code: 05H, DCR C Code: 0DH, DCR D Code: 15H, DCR E Code: 1DH, DCR H Code: 25H, DCR L Code: 2DH

Durch die Ausführung dieses Befehls wird der Inhalt des angegebenen Registers um eins erniedrigt.

DCR M Code: 35H

Bei Ausführung dieses Befehls wird der Inhalt der Speicherzelle, deren Adresse durch den Inhalt des Registerpaars HL bestimmt ist, um eins erniedrigt (siehe Beispiel bei INR M). Es werden alle Zustandsbits außer dem Übertragsbit beeinflußt.

CMA

Komplementiere den Inhalt des Akkumulators
(complement accumulator)

Code: 2FH

Jedes Bit des Inhalts des Akkumulators wird komplementiert. Wenn an einer Stelle der Wert 1 stand, so steht nach der Ausführung dieses Befehls an dieser Stelle der Wert 0. Stand vorher der Wert 0 an dieser Stelle, dann steht nachher dort der Wert 1.

Die Zustandsbits werden nicht verändert.

DAA

Führe eine Dezimalkorrektur aus
(decimal adjust accumulator)

Code: 27H

Der Inhalt des Akkumulators, der eine 8-bit-Dualzahl darstellt, wird nach Ausführung dieser Operation in zwei 4-bit-BCD-Zahlen korrigiert.

Abb. 8.2-1 zeigt anhand eines Flußdiagramms den prinzipiellen Ablauf.

Es werden alle Zustandsbits verändert.

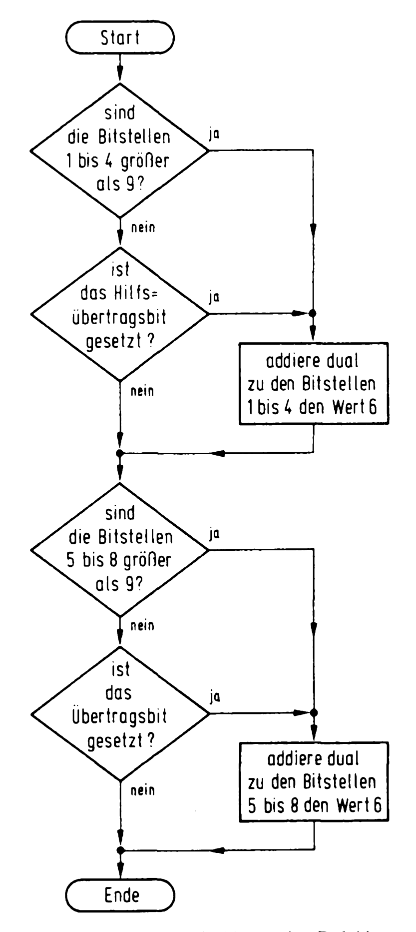


Abb. 8.2-1 Flußdiagramm zur Erklärung des Befehls DAA

3. Befehle, die eine Nulloperation darstellen
NOP

Keine Operation (no operation)

Code: 00H

Es wird bei der Ausführung dieses Befehls keine Operation durchgeführt.

Die Zustandsbits werden nicht verändert. Man verwendet diesen Befehl zum Beispiel für Verzögerungen.

4. Datenübertragungsbefehle
(data transfer instructions)
MOV...

Kopiere Registerinhalt (bzw. Speicherinhalt) nach Registerinhalt (bzw. Speicherinhalt) (move)

MOV A,A Code: 7FH, MOV A,B Code: 78H,
MOV A,C Code: 79H, MOV A,D Code: 7AH,
MOV A,E Code: 7BH, MOV A,H Code: 7CH,
MOV A,L Code: 7DH, MOV A,M Code: 7EH.
MOV B,A Code: 47H, MOV B,B Code: 40 H,
MOV B,C Code: 41H, MOV B,D Code: 42H,
MOV B,E Code: 43H, MOV B,H Code: 44H,
MOV B,L Code: 45H, MOV B,M Code: 46H.
MOV C,A Code: 4FH, MOV C,B Code: 48H,
MOV C,C Code: 49H, MOV C,D Code: 4AH,
MOV C,E Code: 4BH, MOV C,H Code: 4CH,
MOV C,I Code: 4DH, MOV C,M Code: 4EH.
MOV D,A Code: 57H, MOV D,B Code: 50H,
MOV D,C Code: 51H, MOV D,D Code: 52H,
MOV D,E Code: 53H, MOV D,H Code: 54H,
MOV D,L Code: 55H, MOV D,M Code: 56H.
MOV E,A Code: 5FH, MOV E,B Code: 58H,
MOV E,C Code: 59H, MOV E,D Code: 5AH,
MOV E,E Code: 5BH, MOV E,H Code: 5CH,
MOV E,L Code: 5DH, MOV E,M Code: 5EH.
MOV H,A Code: 67H, MOV H,B Code: 60H,
MOV H,C Code: 61H, MOV H,D Code: 62H,
MOV H,E Code: 63H, MOV H,H Code: 64H,
MOV H,L Code: 65H, MOV H,M Code: 66H.
MOV L,A Code: 6FH, MOV L,B Code: 68H,
MOV L,C Code: 69H, MOV L,D Code: 6AH,
MOV L,E Code: 6BH, MOV L,H Code: 6CH,
MOV L,L Code: 6DH, MOV L,M Code: 6EH.
MOV M,A Code: 77H,
MOV M,B Code: 70H, MOV M,C Code: 71H,
MOV M,D Code: 72H, MOV M,E Code: 73H,
MOV M,H Code: 74H,
MOV M,L Code: 75H.

Der Befehl MOV M,M existiert nicht.

Zur Erklärung wird allgemein geschrieben: MOV x,y. Der Befehl bewirkt bei seiner Ausführung die Kopierung des Inhalts von y nach x. Steht für x oder für y M, so wird wieder eine Speicherzelle verwendet, deren Adresse durch den Inhalt von dem Registerpaar HL bestimmt ist.

Die Zustandsbits werden nicht verändert.

Beispiel:

Der Befehl MOV A,B bedeutet, daß bei seiner Ausführung der Inhalt des Registers B in das Register A befördert wird, ohne den Inhalt des Registers B zu verändern.

Der Befehl MOV C,C z. B. bedeutet eine Nulloperation, da sich nach seiner Ausführung nichts verändert hat.

STAX...

Speichere Akkumulatorinhalt ab
(store accumulator)

STAX B Code: 02H, STAX D Code: 12H.

Hier wird mit den Registerpaaren BC bzw. DE gearbeitet.

Dabei stellt der Inhalt der Registerpaare eine Adresse dar. Für das Registerpaar BC gilt: Der Inhalt des Registers B stellt die höherwertige Stelle der Adresse dar und der Inhalt des Registers C die niederwertige Stelle.

Bei dem Registerpaar DE gilt entsprechend:

Der Inhalt des Registers D stellt die höherwertige Stelle der Adresse dar und der Inhalt des Registers E die niederwertige Stelle.

Bei Ausführung des Befehls wird der Inhalt des Akkumulators in die Speicherzelle kopiert, deren Adresse durch das Registerpaar BC (bei STAX B) bzw. durch das Registerpaar DE (bei STAX D) gegeben ist. Die Zustandsbits werden nicht verändert.

LDAX...
Lade den Akkumulator
(load accumulator)

LDAX B Code: 0AH, LDAX D Code: 1AH.
Bei der Ausführung dieses Befehls wird der Inhalt der Speicherzelle in den Akkumulator übertragen, die durch den Inhalt des betreffenden Registerpaars bestimmt ist (siehe auch Erklärung des STAX...-Befehls).

Die Zustandsbits werden nicht verändert.

5. Register oder Speicher nach Akkumulator-Befehle (register or memory to accumulator instructions)

ADD...
Addiere Registerinhalt oder Inhalt der durch HL bestimmten Speicherzelle zu dem Inhalt des Akkumulators (add register or memory to accumulator).

ADD A Code: 87H, ADD B Code: 80H,
ADD C Code: 81H, ADD D Code: 82H,
ADD E Code: 83H, ADD H Code: 84H,
ADD L Code: 85H, ADD M Code: 86H.

Der Inhalt des angegebenen Registers wird zum Inhalt des Akkumulators addiert und das Ergebnis wird im Akkumulator abgespeichert.

Bei ADD M wird entsprechend der Inhalt der durch HL bestimmten Speicherzelle zum Inhalt des Akkumulators addiert und das Ergebnis wird im Akkumulator abgespeichert.

Das vor der Ausführung des Befehls vorhandene Übertragsbit wird bei Ausführung dieser Operation nicht berücksichtigt.

Es werden alle Zustandsbits verändert.

ADC...
Addiere Registerinhalt oder Inhalt der durch HL bestimmten Speicherzelle zum Inhalt des Akkumulators mit Berücksichtigung des Übertragsbits (add register or memory to accumulator with carry).

ADC A Code: 8FH, ADC B Code: 88H,
ADC C Code: 89H, ADC D Code: 8AH,
ADC E Code: 8BH, ADC H Code: 8CH,
ADC L Code: 8DH, ADC M Code: 8EH,
Dieser Befehl ist mit dem vorhergehenden Befehl identisch, nur daß nun das Übertragsbit mitberücksichtigt wird. Der Inhalt des Übertragsbits wird nämlich bei der Ausführung des Befehls zum Inhalt des Akkumulators hinzugezählt (dabei wird dieses Bit in ein 8-bit-Wort verwandelt, so daß es die letzte, niederwertigste Stelle bildet).

Es werden alle Zustandsbits verändert.

SUB...
Subtrahiere Registerinhalt oder Inhalt der durch HL bestimmten Speicherzelle vom Inhalt des Akkumulators (subtract register or memory from accumulator).

SUB A Code: 97H, SUB B Code: 90H,
SUB C Code: 91H, SUB D Code: 92H,
SUB E Code: 93H, SUB H Code: 94H,
SUB L Code: 95H, SUB M Code: 96H.
Dieser Befehl bewirkt eine Subtraktion des Inhalts des angegebenen Registers oder der Speicherzelle (durch HL bestimmt) von dem Inhalt des Akkumulators und eine Abspeicherung des Ergebnisses in dem Akkumulator. Das Übertragsbit bleibt dabei unberücksichtigt.

Die Zustandsbits werden alle verändert.

SBB...
Subtrahiere Registerinhalt oder Inhalt der durch HL bestimmten Speicherzelle vom Inhalt des Akkumulators unter Berücksichtigung des Übertragsbits (subtract register or memory from accumulator with borrow).

SBB A Code: 9FH, SBB B Code: 98H, SBB C Code: 99H, SBB D Code: 9AH, SBB E Code: 9BH, SBB H Code: 9CH, SBB L Code: 9DH, SBB M Code: 9EH.

Dieser Befehl arbeitet genauso wie der vorhergehende, nur daß das Übertragsbit mit berücksichtigt wird.

Alle Zustandsbits werden verändert.

ANA...

Wende die logische UND-Verknüpfung auf den Registerinhalt oder die Speicherzelle (deren Adresse durch HL bestimmt ist) und den Inhalt des Akkumulators an (logical AND-register or memory with accumulator).

ANA A Code: A7H, ANA B Code: A0H, ANA C Code: A1H, ANA D Code: A2H, ANA E Code: A3H, ANA H Code: A4H, ANA L Code: A5H, ANA M Code: A6H.

Der Inhalt des angegebenen Registers oder der durch HL bestimmten Speicherzelle wird mit dem Inhalt des Akkumulators durch UND verknüpft.

Das Ergebnis wird im Akkumulator abgelegt. Die UND-Verknüpfung geschieht dabei Bitstelle für Bitstelle.

Alle Zustandsbits außer dem Hilfsübertragsbit werden verändert.

XRA...

Wende die logische EXKLUSIV-ODER-Verknüpfung auf den Registerinhalt oder Speicherzelle (deren Adresse durch HL bestimmt ist) und den Inhalt des Akkumulators an (logical exclusiv-or register or memory with accumulator).

XRA A Code: AFH, XRA B Code: A8H, XRA C Code: A9H, XRA D-Code: AAH, XRA E Code: ABH, XRA H Code: ACH, XRA L Code: ADH, XRA M Code: AEH.

Der Inhalt des angegebenen Registers oder der durch HL bestimmten Speicherzelle wird mit dem Inhalt des Akkumulators Exklusiv-ODER verknüpft.

Das Ergebnis wird im Akkumulator gespeichert. Das Übertragsbit wird stets zurückgesetzt (auf 0).

Alle Zustandsbits werden verändert (außer dem Hilfsübertragsbit).

ORA...

Wende die logische ODER-Verknüpfung auf den Registerinhalt oder die Speicherzelle (deren Adresse durch das Registerpaar HL bestimmt ist) und den Inhalt des Akkumulators an (logical or register or memory with accumulator).

ORA A Code: B7H, ORA B Code: B0H, ORA C Code: B1H, ORA D Code: B2H, ORA E Code: B3H, ORA H Code: B4H, ORA L Code: B5H, ORA M Code: B6H,

Der Inhalt des angegebenen Registers oder der durch HL bestimmten Speicherzelle wird mit dem Inhalt des Akkumulators ODER verknüpft.

Das Ergebnis wird im Akkumulator gespeichert.

Alle Zustandsbits werden verändert (außer dem Hilfsübertragsbit).

CMP...

Vergleiche den Registerinhalt oder den Speicherzelleninhalt (deren Adresse durch das Registerpaar HL bestimmt ist) mit dem Inhalt des Akkumulators (compare register or memory with accumulator).

CMP A Code: BFH, CMP B Code: B8H, CMP C Code: B9H, CMP D Code: BAH, CMP E Code: BBH, CMP H Code: BCH, CMP L Code, BDH, CMP M Code: BEH.

Der Inhalt des angegebenen Registers oder der durch HL bestimmten Speicherzelle wird vom Inhalt des Akkumulators intern subtrahiert und die Zustandsbits werden entsprechend dem Ergebnis dieser Subtraktion verändert. Der Inhalt des angegebenen Registers bzw. Speicherzelle sowie der Inhalt des Akkumulators bleiben dabei unverändert.

Alle Zustandsbits werden verändert.

6. „Rotiere-Akkumulator"-Befehle (rotate accumulator instructions)

RLC

Rotiere den Inhalt des Akkumulators nach links (rotate accumulator left).

Code: 07H

Der Inhalt des Akkumulators wird bei der Ausführung dieses Befehls um ein bit nach links verschoben. Das Bit 8 (höherwertiges Bit), das eigentlich verloren ginge, wird als Bit 1 (niederwertiges Bit) wieder eingeschoben. Gleichzeitig nimmt das Übertragsbit den gleichen Zustand an, den vorher das Bit 8 hatte.

Nur das Übertragsbit wird verändert.

RRC

Rotiere den Inhalt des Akkumulators nach rechts (rotate accumulator right).

Code: 0FH

Bei Ausführung dieses Befehls wird der Inhalt des Akkumulators um ein bit nach rechts verschoben. Das Bit 1 wird an die Stelle des Bit 8 gesetzt und wird gleichzeitig vom Übertragsbit übernommen.

RAL

Rotiere den Inhalt des Akkumulators nach links durch das Übertragsbit (rotate accumulator left through carry).

Code: 17H

Der Inhalt des Akkumulators wird um ein bit nach links verschoben. Dabei gelangt das Bit 8 in das Übertragsbit und zuvor gelangt das Übertragsbit an die Stelle des Bit 1.

Auch hier wird nur das Übertragsbit beeinflußt.

RAR

Rotiere den Inhalt des Akkumulators nach rechts durch das Übertragsbit (rotate accumulator right through carry).

Code: 1FH

Der Inhalt des Akkumulators wird um ein Bit nach rechts geschoben. Dabei gelangt das Bit 1 in das Übertragsbit und zuvor gelangt das Übertragsbit an die Stelle des Bit 8 im Akkumulator.

Nur das Übertragsbit wird verändert.

7. Registerpaarbefehle (register pair instructions)

PUSH...

Schiebe Daten auf Stack (push data on stack).

PUSH B Code: C5H, PUSH D Code: D5H, PUSH H Code: E5H

Der Inhalt des angegebenen Registerpaars (BC oder DE oder HL) wird in zwei Bytes in der Speicherzelle auf den Adressen abgespeichert, die durch das Registerpaar SP bestimmt sind. Dabei wird z. B. bei Ausführung des Befehls PUSH B der Inhalt des Registers B in der Speicherzelle abgespeichert, deren Adresse um eins geringer ist, als die Adresse, die durch den Inhalt des Registerpaars SP gegeben ist. Der Inhalt des Registers C wird in der Speicherzelle abgespeichert, deren Adresse um zwei geringer ist, als die Adresse, die durch den Inhalt des Registerpaars SP angegeben ist. Anschließend wird vom Inhalt des Registerpaars SP der Wert 2 abgezogen und das Ergebnis im Registerpaar SP gespeichert.

Die Zustandsbits werden nicht verändert.

PUSH PSW Code: F5H

Bei Ausführung dieses Befehls werden die Inhalte der Zustandsbits und der Inhalt des Akkumulators abgespeichert. *Tabelle 8.2-1* zeigt das Zuordnungsschema der einzelnen Zustandsbits zu einem Byte.

Die Inhalte der Zustandsbits werden in der dargestellten Byteform in die Speicherzelle übertragen, deren Adresse um eins niedriger ist, als sie durch den Inhalt des Registerpaars SP gegeben ist.

Der Inhalt des Akkumulators wird in der Speicherzelle abgespeichert, deren Adresse um zwei niedriger ist, als die durch den Inhalt des Registerpaars SP gegebene. Der Inhalt des

Registerpaars SP wird anschließend, wie im obigen Beispiel, um zwei erniedrigt.

Die Zwischenbits werden nicht verändert.

POP...

Hole die Daten von dem Stack (pop data off stack).

POP B Code: C1H, POP D Code: D1H, POP H Code: E1H, POP PSW Code: F1H.

Die Ausführung dieses Befehls stellt die genaue Umkehrung des PUSH-Befehls dar. Die Daten werden wieder von den Speicherzellen in die Register geladen.

(Für die Codierung der Zustandsbits gilt selbstverständlich das gleiche Schema wie bei dem Befehl PUSH PSW.) Der Inhalt des Registerpaars SP wird am Ende der Ausführung des Befehls um zwei erhöht. Bei dem Befehl POP D z. B. wird bei dessen Ausführung zunächst das Register E von der Speicherzelle geladen, deren Adresse durch den Inhalt des Registerpaars SP bestimmt ist. Dann wird Register D von der Speicherzelle geladen, deren Adresse durch den um eins erhöhten Inhalt des Registerpaars SP bestimmt ist.

Anschließend wird der Inhalt des Registerpaars SP, wie schon gesagt, um zwei erhöht.

Die Zustandsbits werden nur bei der Ausführung des Befehls POP PSW verändert.

DAD...

doppelt genaue Addition (double add).

DAD B Code: 09H, DAD D Code: 19H, DAD H Code: 29H, DAD SP Code: 39H.

Bei Ausführung dieses Befehls wird die 16-bit-Zahl, die durch den Inhalt des angegebenen Speicherpaars bestimmt ist, zu dem Inhalt des Registerpaars HL addiert. Das Ergebnis wird in dem Registerpaar HL abgespeichert.

Es wird nur das Übertragsbit verändert.

INX...

Erhöhe den Inhalt des angegebenen Registerpaars um eins (increment register pair).

INX B Code: 03H, INX D Code: 13H, INX H Code: 23H, INX SP Code: 33H.

Bei der Ausführung dieses Befehls wird der Inhalt (16-bit-Zahl) des angegebenen Registerpaars um eins erhöht.

Die Zustandsbits werden nicht verändert.

DCX...

Erniedrige den Inhalt des angegebenen Registerpaars um eins (decrement register pair).

DCX B Code: 0BH, DCX D Code: 1BH, DCX H Code: 2BH, DCX SP Code: 3BH.

Bei Ausführung dieses Befehls wird der Inhalt (16-bit-Zahl) des angegebenen Registerpaars um eins erniedrigt.

Die Zustandsbits werden nicht verändert.

XCHG

Vertausche die Registerinhalte (exchange registers).

Code: EBH

Der Inhalt des Registerpaars HL wird mit dem Inhalt des Registerpaars DE vertauscht.

Dabei gelangt der Inhalt des Registers H in das Register D und der ursprüngliche Inhalt des Registers D gelangt in das Register H.

Der Inhalt des Registers L gelangt in das Register E und der ursprüngliche Inhalt des Registers E gelangt in das Register L.

Die Zustandsbits werden nicht verändert.

Tabelle 8.2-1

Bit-Nummer	8	7	6	5	4	3	2	1
Inhalt	Vorzeichenbit	Nullbit	0	Hilfsübertragsbit	0	Paritätsbit	1	Übertragsbit

XTHL
Vertausche Stack (exchange stack).

Code: E3H
Der Inhalt des Registers L wird mit dem Inhalt der Speicherzelle vertauscht, deren Adresse durch den Inhalt des Registerpaars SP bestimmt ist.

Der Inhalt des Registers H wird mit dem Inhalt der Speicherzelle vertauscht, deren Adresse durch den um eins erhöhten Inhalt des Registerpaars SP bestimmt ist.

Die Zustandsbits werden nicht verändert.

SPHL
Lade das Registerpaar SP von dem Registerpaar HL (load SP from HL).

Code: F9H
Der Inhalt des Registerpaars HL wird in das Registerpaar SP übertragen. Der Inhalt des Registerpaars HL bleibt dabei unverändert.

Die Zustandsbits werden nicht verändert.

8. Folge-Instruktionen (immediate instructions)

LXI...
Lade das angegebene Registerpaar mit den nachfolgenden zwei Bytes.

LXI B Code: 01H, LXI D Code: 11H, LXI H Code: 21H, LXI SP Code: 31H.
In das angegebene Registerpaar werden die nach der Instruktion folgenden zwei Bytes geladen: Z. B. der Befehl LXI D, 1200H, der in der Maschinensprache folgendermaßen aussieht: 11 00 12. 11 stellt dabei den Code für den Befehl „LXI D" dar. Die folgenden zwei Bytes beinhalten die Dateninformation. Nach Ausführen des Befehls enthält das Register E den Wert 00H und das Register D den Wert 12H. Bei der Übersetzung des Befehls LXI D, 1200H in den Maschinencode ist darauf zu achten, daß das höherwertige Byte erst als drittes Byte in dem entsprechenden Maschinencode auftritt. Das niederwertige Byte erscheint als zweites Byte in dem Maschinencode.

Es wird bei Ausführung des Befehls das direkt der Instruktion folgende Byte in das niederwertige Register (im Beispiel das Register E) eingespeichert und das nächste Byte in das höherwertige Register (also im Beispiel das Register D).

Niederwertige Register sind: Register C, E, L.
Höherwertige Register sind: Register B, D, H.

Die Zustandsbits werden nicht verändert.

MVI...
Speichere das der Instruktion folgende Datenbyte (move immediate data).

MVI A Code: 3EH, MVI B Code: 06H, MVI C Code: 0EH, MVI D Code: 16H, MVI E Code: 1EH, MVI H Code: 26H, MVI L Code: 2EH, MVI M Code: 36H.
In das angegebene Register oder in die durch HL bestimmte Speicherzelle wird das der Instruktion folgende Byte geladen.

Die Zustandsbits werden nicht verändert.

ADI
Addiere das der Instruktion folgende Byte zum Inhalt des Akkumulators (add immediate to accumulator).

Code: C6H
Das der Instruktion folgende Byte wird zum Inhalt des Akkumulators addiert und das Ergebnis wird im Akkumulator abgespeichert.

Alle Zustandsbits werden verändert.

ACI
Addiere das der Instruktion folgende Byte zum Inhalt des Akkumulators unter Berücksichtigung des Zustands des Übertragsbits (add immediate to accumulator with carry).

Code: CEH
Das der Instruktion folgende Byte wird zum Inhalt des Akkumulators addiert mit Berücksichtigung des Übertragsbits und das Ergebnis wird im Akkumulator abgespeichert (siehe auch ADC...-Befehl).

Alle Zustandsbits werden verändert.

SUI
Subtrahiere das der Instruktion folgende Byte vom Inhalt des Akkumulators (subtract immediate from accumulator).

Code: D6H
Das der Instruktion folgende Byte wird vom Inhalt des Akkumulators subtrahiert und das Ergebnis wird im Akkumulator abgespeichert.
Alle Zustandsbits werden verändert.

SBI
Subtrahiere das der Instruktion folgende Byte vom Inhalt des Akkumulators unter Berücksichtigung des Zustands des Übertragsbits (subtract immediate from accumulator with borrow).

Code: DEH
Das der Instruktion folgende Byte wird vom Inhalt des Akkumulators subtrahiert unter der Berücksichtigung des Übertragsbits. Das Ergebnis wird im Akkumulator abgespeichert.
Alle Zustandsbits werden verändert.

ANI
Wende die logische UND-Verknüpfung auf das der Instruktion folgende Byte und den Inhalt des Akkumulators an (and immediate with accumulator).

Code: E6H
Das der Instruktion folgende Byte wird mit dem Inhalt des Akkumulators UND verknüpft und das Ergebnis wird im Akkumulator gespeichert. Das Übertragsbit wird auf Null gesetzt.
Alle Zustandsbits außer dem Hilfsübertragsbit werden verändert.

XRI
Wende die logische Exklusiv-ODER-Verknüpfung auf das der Instruktion folgende Byte und den Inhalt des Akkumulators an (exclusiv or immediate with accumulator).

Code: EEH
Das der Instruktion folgende Byte wird mit dem Inhalt des Akkumulators Exklusiv-ODER verknüpft und das Ergebnis wird im Akkumulator abgespeichert.
Alle Zustandsbits außer dem Hilfs-Übertragsbit werden verändert. (Das Übertragsbit wird auf Null gesetzt.)

ORI
Wende die logische ODER-Verknüpfung auf das der Instruktion folgende Byte und den Inhalt des Akkumulators an (or immediate with accumulator).

Code: F6H
Das der Instruktion folgende Byte wird mit dem Inhalt des Akkumulators ODER verknüpft und das Ergebnis wird im Akkumulator abgespeichert.
Das Übertragsbit wird auf Null gesetzt.
Alle Zustandsbit außer dem Hilfsübertragsbit werden verändert.

CPI
Vergleiche das der Instruktion folgende Byte mit dem Inhalt des Akkumulators (compare immediate with accumulator).

Code: FEH
Das der Instruktion folgende Byte wird mit dem Inhalt des Akkumulators verglichen, indem das Byte intern vom Inhalt des Akkumulators subtrahiert wird, ohne den Inhalt des Akkumulators zu verändern.
Alle Zustandsbits werden verändert.

9. Direkte Andressierungs-Befehle (direkt addressing instructions)
STA
Speichere den Inhalt des Akkumulators direkt in die angegebene Speicherzelle (store accumulator direct).

Code: 32H
Der Inhalt des Akkumulators wird in die Speicherzelle gespeichert, deren Adresse durch die

der Instruktion folgenden zwei Bytes bestimmt ist. Dabei bildet das der Instruktion direkt folgende Byte die niederwertige Stelle der Adresse und das nächste folgende Byte die höherwertige Adresse.

Beispiel: Es wird der Befehl STA 1213H betrachtet. Umgesetzt in den Maschinencode lautet er: 32 13 12. Bei Ausführung des Befehls wird der Inhalt des Akkumulators in die Speicherzelle mit der Adresse 1213H gespeichert.

Die Zustandsbits werden nicht verändert.

LDA
Lade den Akkumulator direkt mit dem Inhalt der angegebenen Speicherzelle (load accumulator direct).

Code: 3AH
Der Inhalt der Speicherzelle, deren Adresse durch die der Instruktion folgenden zwei Bytes bestimmt ist, wird in den Akkumulator gespeichert.

Dabei bildet das der Instruktion folgende Byte die niederwertige Stelle der Adresse und das nächste Byte die höherwertige Stelle der Adresse.

Die Zustandsbits werden nicht verändert.

SHLD
Speichere den Inhalt des Registerpaares HL direkt auf die Speicherzellen, die angegeben sind (store H and L direct).

Code: 22H
Der Inhalt des Registers L wird in die Speicherzelle gespeichert, die durch die der Instruktion folgenden zwei Bytes bestimmt ist. Das direkt der Instruktion folgende Byte bildet die niederwertige Stelle der Adresse und das nächste Byte die höherwertige Stelle der Adresse. Der Inhalt des Registers H wird in die Speicherzelle gespeichert, deren Adresse die um eins erhöhte Adresse darstellt, die durch die der Instruktion folgenden zwei Bytes bestimmt ist.

Die Zustandsbits werden nicht verändert.

LHLD
Der Inhalt der zwei Speicherzellen, die angegeben sind, wird in das Registerpaar HL gespeichert (load H and L direct).

Code: 2AH
Der Inhalt der Speicherzelle, deren Adresse durch die der Instruktion folgenden zwei Bytes bestimmt ist, wird in das Register L gespeichert.

Der Inhalt der Speicherzelle, deren Adresse die um eins erhöhte Adresse darstellt, die durch die der Instruktion folgenden zwei Bytes bestimmt ist, wird in das Register H gespeichert.

Die Zustandsbits werden nicht verändert.

10. Sprung-Befehle (jump instructions)

PCHL

Lade den Programmzähler (load programm counter).

Code: E9
Der Inhalt des Registers H ersetzt die höherwertige Stelle des Programmzählers und der Inhalt des Registers L ersetzt die niederwertige Stelle des Programmzählers.

Die Zustandsbits werden nicht verändert.

JMP
Springe (jump).

Code: C3H
Die Ausführung des Programms setzt sich bei der Adresse der Speicherzelle fort, deren Adresse durch die der Instruktion folgenden zwei Bytes bestimmt ist. Das der Instruktion folgende Byte bestimmt dabei die niederwertige Stelle der Adresse und das nächste Byte bestimmt die höherwertige Stelle der Adresse.

Die Zustandsbits werden nicht verändert.

JC
Springe, wenn das Übertragsbit gesetzt ist (jump if carry).

Code: DAH
Wenn der Inhalt des Übertragsbits 1 ist, erfolgt die Fortsetzung der Ausführung des Programms an der Stelle des Speichers, die genauso wie bei der JMP-Instruktion bestimmt wird.
Die Zustandsbits werden nicht verändert.

JNC
Springe, wenn das Übertragsbit nicht gesetzt ist (jump if not carry).

Code: D2H
Wenn der Inhalt des Übertragsbit gleich 0 ist, erfolgt ein Sprung wie bei der JMP-Instruktion.
Die Zustandsbits werden nicht verändert.

JZ
Springe, wenn das Nullbit gesetzt ist (jump if zero).

Code: CAH
Wenn der Inhalt des Nullbits gleich 1 ist, erfolgt ein Sprung wie bei der JMP-Instruktion.
Die Zustandsbits werden nicht verändert.

JNZ
Springe, wenn das Nullbit nicht gesetzt ist (jump if not zero).

Code: C2H
Wenn der Inhalt des Nullbits gleich 0 ist, erfolgt ein Sprung wie bei der JMP-Instruktion.
Die Zustandsbits werden nicht verändert.

JM
Springe, wenn das Vorzeichenbit gesetzt ist (jump if minus).

Code: FAH
Wenn der Inhalt des Vorzeichenbits gleich 1 ist, erfolgt ein Sprung wie bei der JMP-Instruktion.
Die Zustandsbits werden nicht verändert.

JP
Springe, wenn das Vorzeichenbit nicht gesetzt ist (jump if positiv).

Code: F2H
Wenn der Inhalt des Vorzeichenbits gleich 0 ist, erfolgt ein Sprung wie bei der JMP-Instruktion.
Die Zustandsbits werden nicht verändert.

JPE
Springe, wenn das Paritätsbit gesetzt ist (jump if parity even).

Code: EAH
Wenn der Inhalt des Paritätsbits gleich 1 ist, erfolgt ein Sprung wie bei der JMP-Instruktion.
Die Zustandsbits werden nicht verändert.

JPO
Springe, wenn das Paritätsbit nicht gesetzt ist (jump if parity odd).

Code: E2H
Wenn der Inhalt des Paritätsbits gleich 0 ist, erfolgt ein Sprung wie bei der JMP-Instruktion.
Die Zustandsbits werden nicht verändert.

11. Unterprogrammaufruf-Befehle (call instructions)

Mit dieser Gruppe von Befehlen ist es möglich, Unterprogramme zu verwirklichen. Wird ein Programmteil häufig in gleicher Weise in einem Programm benötigt, so ist es meist vorteilhaft, ein solches Programmteil als Unterprogramm auszuführen. Dabei wird dieses Programmteil nur einmal geschrieben. Vom Hauptprogramm aus erfolgt dann ein Unterprogramm-Aufruf, der dieses Programmteil ablaufen läßt. Nach Ausführung des Unterprogramms erfolgt ein sogenannter Rücksprung. Dadurch wird das Hauptprogramm an der Stelle weiter ausgeführt, an der der Unterprogrammsprung erfolgte. Es ist nun möglich, das gleiche Unterprogramm von verschiedenen Stellen des Hauptprogramms aus aufzurufen und so mehrmals zu verwenden.

CALL
Rufe auf (call).

Code: CDH
Zunächst wird, wie bei allen Unterprogrammbefehlen, der Inhalt des Programmzählers festgehalten, um einen Rücksprung zum rufenden Programm zu ermöglichen. Es wird dabei der Inhalt des Programmzählers festgehalten, der die Adresse der Speicherzelle beinhaltet, die den nächsten Befehl enthält, der nach dem CALL-Befehl folgt.

Die höherwertige Stelle des Programmzählers wird dabei in die Speicherzelle gespeichert, deren Adresse durch den um eins erniedrigten Inhalt des Registerpaars SP bestimmt wird, und die niederwertige Stelle des Programmzählers wird in die Speicherzelle gespeichert, deren Adresse durch den um zwei erniedrigten Inhalt des Registerpaars SP bestimmt ist. Anschließend wird der Inhalt des Registerpaars SP um zwei erniedrigt.

Die Fortsetzung des Programms erfolgt an der Stelle des Speichers, deren Adresse durch die nachfolgenden zwei Bytes der Instruktion bestimmt ist (siehe auch JMP-Instruktion).

Die Zustandsbits werden nicht verändert.

CC
Rufe auf, wenn das Übertragsbit gesetzt ist (call if carry).

Code: DCH
Wenn der Inhalt des Übertragsbits gleich 1 ist, erfolgt die Ausführung eines CALL-Befehls.

CNC
Rufe auf, wenn das Übertragsbit nicht gesetzt ist (call if no carry).

Code: D4H
Wenn der Inhalt des Übertragsbits gleich 0 ist, erfolgt die Ausführung eines CALL-Befehls.

CZ
Rufe auf, wenn das Nullbit gesetzt ist (call if zero).

Code: CCH
Wenn der Inhalt des Nullbits gleich 1 ist, erfolgt die Ausführung eines CALL-Befehls.

CNZ
Rufe auf, wenn das Nullbit nicht gesetzt ist (call if not zero).

Code: C4H
Wenn der Inhalt des Nullbits gleich 0 ist, erfolgt die Ausführung eines CALL-Befehls.

CM
Rufe auf, wenn das Vorzeichenbit gesetzt ist (call if minus).

Code: FCH
Wenn der Inhalt des Vorzeichenbits gleich 1 ist, erfolgt die Ausführung eines CALL-Befehls.

CP'
Rufe auf, wenn das Vorzeichenbit nicht gesetzt ist (call if positiv).

Code: F4H
Wenn der Inhalt des Vorzeichenbits gleich 0 ist, erfolgt die Ausführung eines CALL-Befehls.

CPE
Rufe auf, wenn das Paritätsbit gesetzt ist (call if parity even).

Code: ECH
Wenn der Inhalt des Paritätsbits gleich 1 ist, erfolgt die Ausführung eines CALL-Befehls.

CPO
Rufe auf, wenn das Paritätsbit nicht gesetzt ist (call if parity odd).

Code E4H
Wenn der Inhalt des Paritätsbits gleich 0 ist, erfolgt die Ausführung eines CALL-Befehls.

12. Befehle für einen Rücksprung von einem Unterprogramm (return from subroutine instructions)

RET
Springe zurück zum rufenden Programm (return).

Code: C9H
Dieser Befehl stellt das Gegenstück zum CALL-Befehl dar. Der Inhalt der Speicherzelle, deren Adresse durch den Inhalt des Registerpaars SP bestimmt ist, wird in die niederwertige Stelle des Programmzählers gespeichert, und der Inhalt der Speicherzelle, deren Adresse den um eins erhöhten Inhalt des Registerpaars darstellt, wird in die höherwertige Stelle des Programmzählers gespeichert. Anschließend wird der Inhalt des Registerpaars SP um zwei erhöht.

Die Zustandsbits werden nicht verändert.

RC
Springe zurück, wenn das Übertragsbit gesetzt ist (return if carry).

Code: D8H
Wenn der Inhalt des Übertragsbits gleich 1 ist, erfolgt die Ausführung eines RET-Befehls.

RNC
Springe zurück, wenn das Übertragsbit nicht gesetzt ist (return if no carry).

Code: D0H
Wenn der Inhalt des Übertragsbits gleich 0 ist, erfolgt die Ausführung eines RET-Befehls.

RZ
Springe zurück, wenn das Nullbit gesetzt ist (return if zero).

Code: C8H
Wenn der Inhalt des Nullbits gleich 1 ist, erfolgt die Ausführung eines RET-Befehls.

RNZ
Springe zurück, wenn das Nullbit nicht gesetzt ist (return if not zero).

Code: C0H
Wenn der Inhalt des Nullbits gleich 0 ist, erfolgt die Ausführung eines RET-Befehls.

RM
Springe zurück, wenn das Vorzeichenbit gesetzt ist (return if minus).

Code: F8H
Wenn der Inhalt des Vorzeichenbits gleich 1 ist, erfolgt die Ausführung eines RET-Befehls.

RP
Springe zurück, wenn das Vorzeichenbit nicht gesetzt ist (return if positiv).

Code: F0H
Wenn der Inhalt des Vorzeichenbits gleich 0 ist, erfolgt die Ausführung eines RET-Befehls.

RPE
Springe zurück, wenn das Paritätsbit gesetzt ist (return if parity even).

Code: E8H
Wenn der Inhalt des Paritätsbits gleich 1 ist, erfolgt die Ausführung eines RET-Befehls.

RPO
Springe zurück, wenn das Paritätsbit nicht gesetzt ist (return if parity odd).

Code: E0H
Wenn der Inhalt des Paritätsbit gleich 0 ist, erfolgt die Ausführung eines RET-Befehls.

13. Befehle für einen „Neustart" (restart instructions)

RST...
Führe einen „Neustart" aus (restart).
RST 0 Code: C7H, RST 1 Code: CFH,
RST 2 Code: D7H, RST 3 Code: DFH,
RST 4 Code: E7H, RST 5 Code: EFH.
RST 6 Code: F7H, RST 7 Code: FFH.
Die Ausführung dieses Befehls entspricht der Ausführung eines CALL-Befehls. Die Adresse, die in den Programmzähler geladen wird, wird aber anders bestimmt. Der Befehl selbst bestimmt diese Adresse. Wird der Befehl RST 0 ausgeführt, so erfolgt der Unterprogrammsprung zur Adresse 0000H.

Bei RST 1 nach 0008H,
bei RST 2 nach 0010H,
bei RST 3 nach 0018H,
bei RST 4 nach 0020H,

bei RST 5 nach 0028H,
bei RST 6 nach 0030H,
bei RST 7 nach 0038H.

14. Befehle für die Unterbrechungsverarbeitung (interrupt instructions)

Die Möglichkeit Unterbrechungen vorzunehmen, stellt bei einem Mikrocomputer einen wichtigen Faktor dar. Es ist damit möglich, durch ein externes Signal den Aufruf eines Unterprogramms zu bewirken. Der Mikroprozessor besitzt dazu einen sogenannten Interrupt-Eingang.

EI

Ermöglichen von Unterbrechungen (enable interrupts).

Code: FBH

Nach Ausführung dieses Befehls können Unterbrechungen durchgeführt werden.

Die Zustandsbits werden nicht verändert.

DI

Keine Möglichkeit für Unterbrechungen (disable interrupts).

Code: F3H

Nach Ausführung dieses Befehls bleiben eventuelle Unterbrechungen ohne Wirkung.

Die Zustandsbits werden nicht verändert.

15. Ein-/Ausgabe-Befehle (input/output instructions)

IN

Eingabe (input).

Code: DBH

Der Inhalt der Peripherieeinheit, deren Adresse durch das der Instruktion folgende Byte gegeben ist, wird in den Akkumulator geladen.

Die Zustandsbits werden nicht verändert.

OUT

Ausgabe (output).

Code: D3H

Der Inhalt des Akkumulators wird in die Peripherieeinheit übertragen, deren Adresse durch das der Instruktion folgende Byte gegeben ist.

Die Zustandsbits werden nicht verändert.

16. Halte-Befehle (halt instructions)

HLT

Halte (halt).

Code: 76H

Der Programmzähler wird noch auf die Adresse der Speicherzelle gesetzt, die den nächsten Befehl enthält. Dann wird der Prozessor angehalten. Durch eine wirksame Unterbrechung (interrupt) oder durch einen Reset (über den Reset-Eingang) des Prozessors kann die Wirkung des Halte-Befehls aufgehoben werden.

Einfaches Beispiel-Programm:

Das Programm hat die Aufgabe, alle möglichen ISO-7-bit-Zeichen auf dem Datensichtgerät auszugeben.

Abb. 8.2-2 zeigt das entsprechende Flußdiagramm.

Nach Start des Programms wird eine Speicherzelle mit dem Namen WERT auf 20H gesetzt. Diese Zahl bestimmt das Zeichen, das zuerst gedruckt wird. Als nächstes wird das Zeichen ausgegeben. Dies geschieht durch Aufruf des Unterprogramms CO. Dieses Unterprogramm befindet sich in dem ROM, der von der Firma Intel mit dem SDK 80 Kit mitgeliefert wird. Dieser ROM, genauer gesagt ein EPROM, enthält ein sogenanntes Monitorprogramm, das es dem Anwender ermöglicht, Programme zu laden, auszuführen und zu korrigieren. Das Unterprogramm CO, das sich auf dem Speicherplatz mit der Adresse 03FAH aufrufen läßt, gibt dem Anwender die Möglichkeit, ein Zeichen auszugeben, das sich in dem Register C befindet. Nach Ausgabe des Zeichens erfolgt nun die Erhöhung der Speicherzelle mit dem Namen WERT um eins. Es

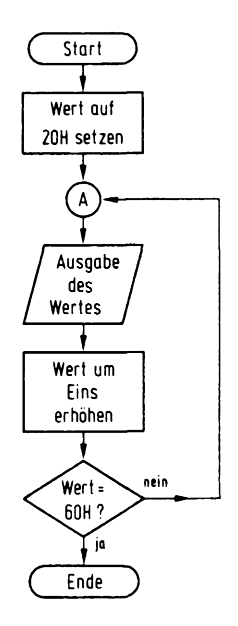


Abb. 8.2-2 Flußdiagramm des Beispielprogramms

Tabelle 8.2-2

Adresse (sedezimal)	Maschinencode	Assemblerbefehl	Kommentar
1300	0E	MVI C,20H	;WERT auf 20H setzen
1301	20		
1302	CD	A: CALL CO	;Zeichen ausgeben
1303	FA		
1304	03		
1305	0C	INR C	;WERT um eins erhöhen
1306	79	MOV A,C	;Zum Vergleich in Akkumulator
1307	FE	CPI 60H	;Mit der Zahl 60H vergleichen
1308	60		
1309	C2	JNZ A	;Falls nicht gleich der Zahl
130A	02		;zur Marke A springen
130B	13		
130C	CF	RST 1	;Aufruf des Monitorprogramms

folgt dann die Abfrage, ob der Inhalt dieser Speicherzelle identisch mit der Zahl 60H ist. Ist dies der Fall, so ist das Programm beendet. Dann erfolgt ein Aufruf des Monitorprogramms mit dem Befehl RST 1. Ist die Bedingung aber nicht erfüllt, so erfolgt ein Sprung zu der Marke A.

Tabelle 8.2-2 zeigt das Programm in der Maschinensprache und in der Assemblersprache.

8.3 Interface für den Mikroprozessor

Abb. 8.3-1 zeigt ein standardmäßiges Interface, das notwendig ist, um ein Mikrocomputersystem zu erstellen. Dieses Interface hat die Aufgabe, die Signale für die Steuerung der Peripherieeinheiten und für die Steuerung des Speichers zu liefern. Ferner beinhaltet dieses Interface einen Taktgenerator, der alle notwendigen Taktsignale für die eigentliche CPU liefert, sowie auch für andere Mikrocomputerkomponenten.

8.4 Speicher RAM und PROM (bzw. EPROM)

Abb. 8.4-1 zeigt ein Blockschaltbild eines kompletten Mikrocomputersystems, wie es von der Firma Intel unter der Bezeichnung SDK 80 Kit als Bausatz mit Platine geliefert wird.

Dieser Bausatz beinhaltet einen RAM-Speicher von 256 Bytes, der auf 1 KByte erweitert werden kann. Ferner sind in dem Bausatz zwei EPROMs mit einer Kapazität von je 1 KByte enthalten, von denen der eine das schon erwähnte Monitorprogramm enthält. Dieses Programm ermöglicht dem Benutzer die Kommunikation mit Hilfe eines Fernschreibers oder des hier beschriebenen Datensichtgerätes mit dem Mikrocomputer.

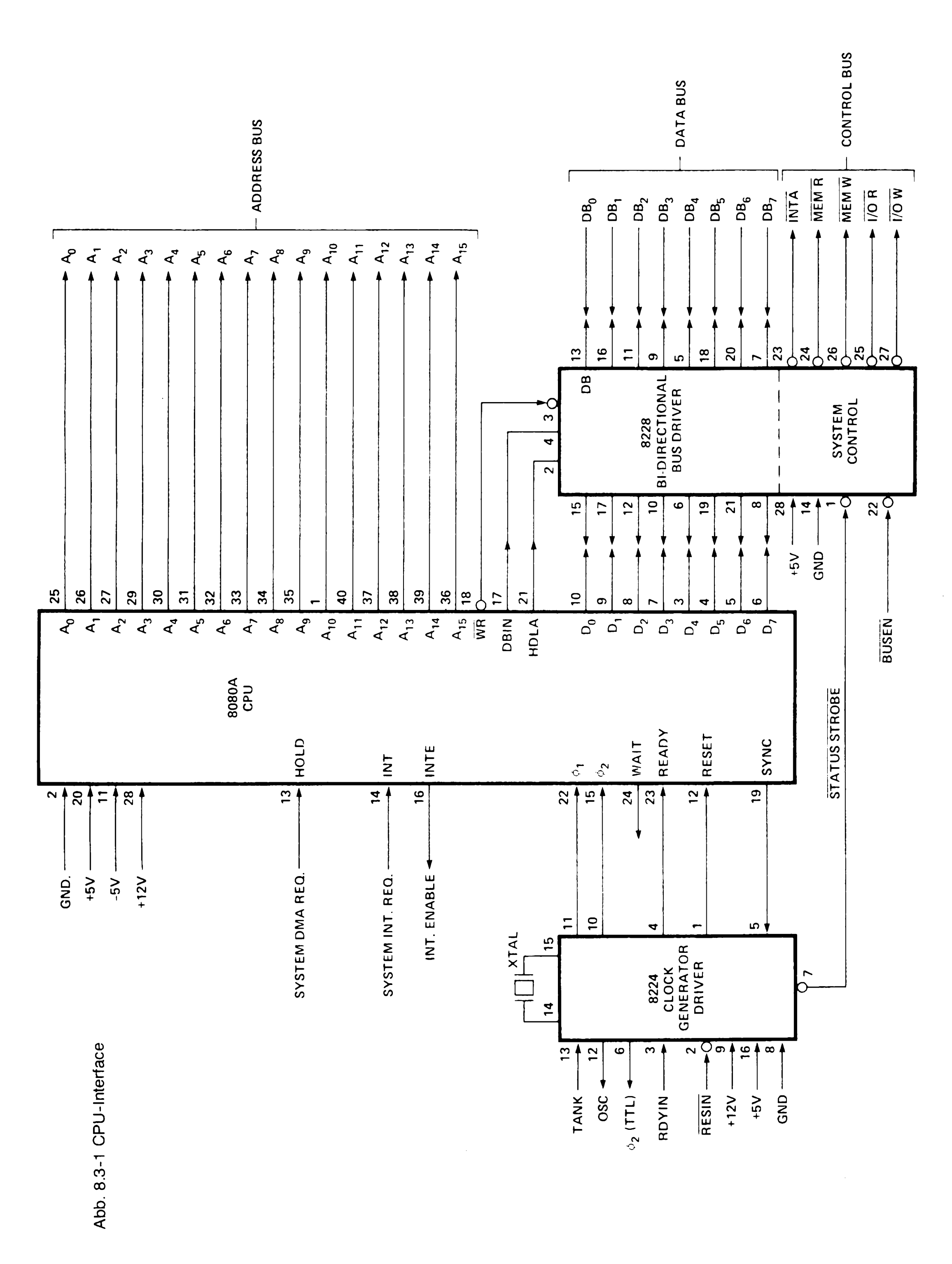


Abb. 8.3-1 CPU-Interface

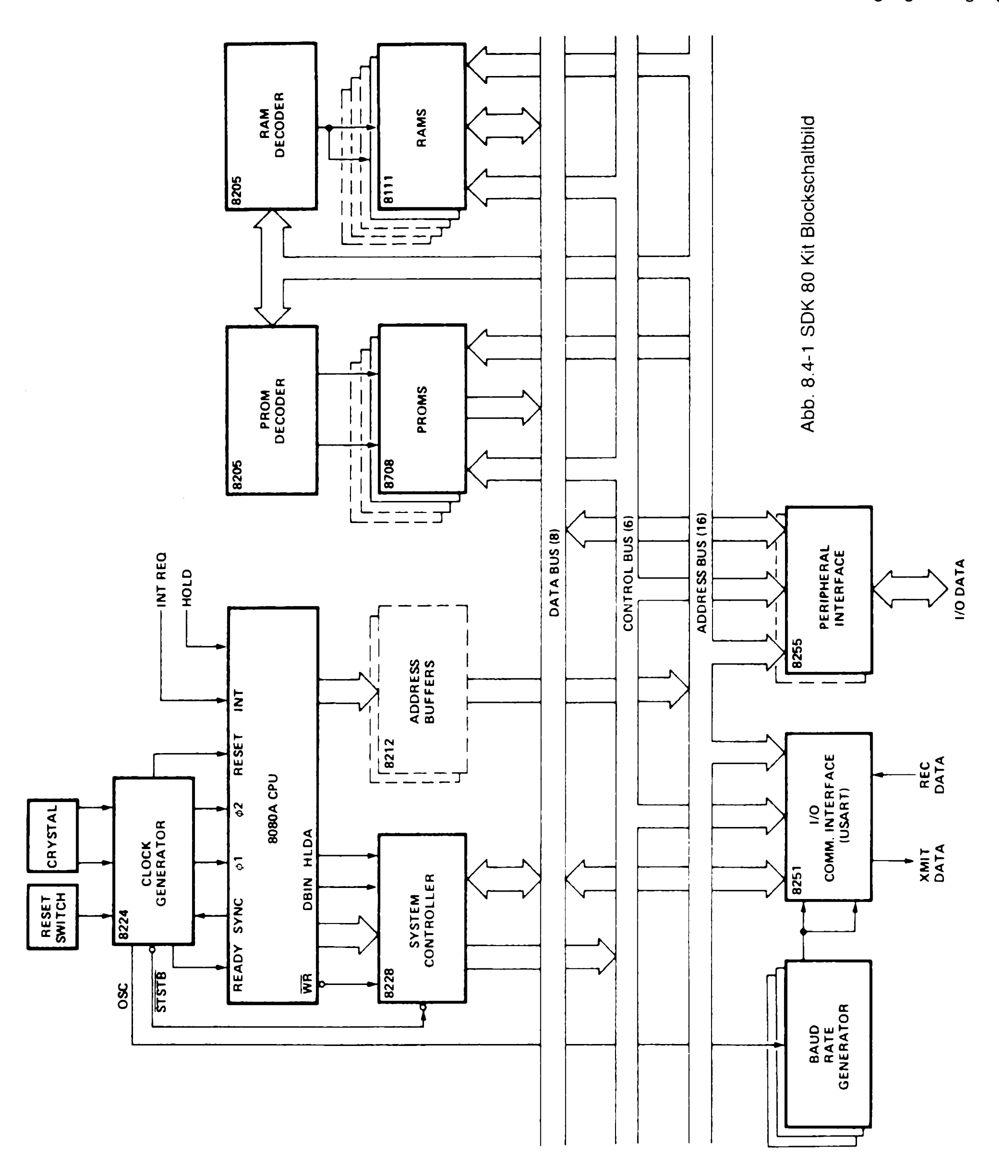


Abb. 8.4-1 SDK 80 Kit Blockschaltbild

8.5 Eingänge/Ausgänge

Der Bausatz SDK 80 Kit ermöglicht den seriellen Datentransport mit Hilfe eines mitgelieferten UARTS (serielles Interface-IC). Der Bausatz enthält ferner noch ein paralleles Interface-IC das programmierbar ist und 24 Datenleitungen nach außen führt. Die Datenleitungen lassen sich in unterschiedlichen Kombinationen zu Ausgängen, Eingängen oder Bus-Leitungen programmieren. Ein zweites derartiges IC kann in die mitgelieferte Platine zusätzlich eingefügt werden.

9 Anwendung und Verknüpfung der aufgebauten Einheiten

Ziel dieses Kapitels ist es, anhand von Anwendungsbeispielen einen Einsatz der Geräte in der Praxis zu ermöglichen.

Dies bezieht sich sowohl auf die Software (Programme etc.) wie auch auf die Hardware (Zusammenschalten der Geräte usw.).

Der Schwerpunkt liegt dabei ganz besonders auf der Software.

Die meisten Programme sind für den Mikrocomputer 8080 ausgelegt, speziell für den SDK 80 Kit.

9.1 Versuch zur Bestimmung der Gravitationskonstante

9.1.1 mit Universalzähler als Grundeinheit

In diesem Abschnitt soll der Gebrauch des Universalzählers, der in Kapitel 3 beschrieben wurde, gezeigt werden. Dazu wird eine einfache Aufgabe gestellt: Die Bestimmung der Gravitationskonstante mit einem Pendel.

Die Formel zur Bestimmung der Gravitationskonstante ergibt sich durch Umstellen der Formel:

$$T = 2\,PI \sqrt{\frac{l}{g}}$$

Man erhält somit:

$$g = \frac{4\,PI^2\,l}{T^2}$$

Als veränderliche Meßgrößen treten also nur l und T auf. Dabei kann l einmal gemessen werden und bleibt dann konstant. Es gilt also nur T elektronisch zu messen.

Abb. 9.1.1-1 zeigt den Aufbau der Meßapparatur. Das Pendel besteht in diesem Fall aus einem dünnen aber nicht dehnbaren Zwirn und einer Pendelmasse, die möglichst klein sein (volumenmäßig) und doch eine große Masse besitzen sollte. Gemessen werden soll die Schwingungsdauer T. Mit Hilfe einer Anordnung der Lichtschranke, wie sie das Bild zeigt, ist es möglich, die Dauer $T/2$ zu messen. Damit der Lichtstrahl sicher unterbrochen wird, ist es nötig, den Zwirn an der betreffenden Stelle mit einem kleinen Röhrchen zu überziehen.

Der mechanische Aufbau kann zum Beispiel mit Fischertechnik gut verwirklicht werden.

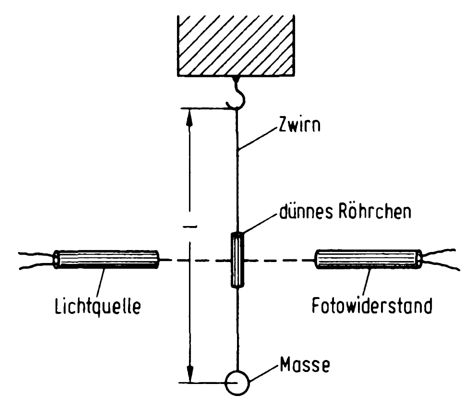


Abb. 9.1.1-1 Versuchsaufbauschema

Mit dem Universalzähler ist es möglich, die Zeit *T/2* zu messen.

Abb. 9.1.1-2 zeigt die Schaltung, wie sie zum Messen der Zeit benötigt wird. Der Ausgang des Fotowiderstandes gelangt dabei an einen Umformer, der z. B. mit dem Fischertechnik-Schaltstab realisiert werden kann. Er kann aber auch diskret aufgebaut werden. Dann folgt ein Relais, das von diesem Umformer angesteuert wird. Anstatt der Schaltung mit einem elektromechanischen Relais kann selbstverständlich auch eine entsprechende Digitalschaltung aufgebaut werden. Das Kontaktprellen des Relais wird anschließend mit zwei NAND-Verknüpfungen unterdrückt. Das entprellte Signal gelangt nun an ein Flipflop. Der Ausgang des Flipflops steuert dabei direkt den Latch-Eingang des Universalzählers. Der Schalter S 1 des Universalzählers wird dazu in Stellung RLC gebracht und der Schalter S 2 in Stellung auto R 0. Über eine NAND-Verknüpfung gelangt ein 100-kHz-Signal immer dann an den Zähleingang des Universalzählers, wenn der Pegel am anderen Eingang derselben NAND-Verknüpfung HIGH ist.

Im folgenden wird nun der Ablauf einer kompletten Messung beschrieben. Es wird angenommen, daß der Pegel am Ausgang Q des Flipflops gerade von HIGH auf LOW überwechselte. Der Universalzähler wird nun rückgesetzt (Programmzähler auf 0). Gleichzeitig wird über den Ausgang Q, der HIGH-Potential führt, die NAND-Verknüpfung befähigt, das 100-kHz-Signal an den Zähleingang durchzuschalten. Wenn das Flipflop umkippt, so wird die NAND-Verknüpfung gesperrt, und es gelangen keine weiteren Impulse an den Zähleingang. Gleichzeitig wird der Rücksetzbefehl aufgehoben und der Universalzähler bzw. die Mikroprogrammsteuereinheit beginnt das Mikroprogramm von der Adresse 0 an abzuarbeiten.

Dabei wird zunächst die Befehlsgruppe „Übertrage Inhalt von Zähler in Anzeigespeicher" ausgeführt. Danach wird der Inhalt des

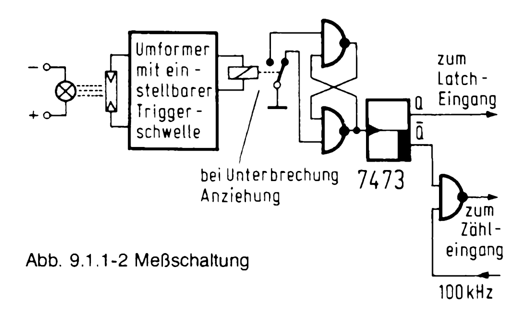


Abb. 9.1.1-2 Meßschaltung

Zählers gelöscht (S 2 in Stellung „auto R 0"). Anschließend wird der Rest des Mikroprogramms abgearbeitet, ohne jedoch Steuervorgänge zu bewirken. Der Bereich des Universalzählers wird mit dem Schalter auf 1x gestellt, so daß bei einer ungefähren Pendellänge von 0,3 m die Abarbeitung des restlichen Mikroprogramms länger dauert, als die Zeit *T/2*. Es könnte sonst passieren, daß bei neuem Beginn der Abarbeitung des Mikroprogramms der Inhalt des Zählers erneut in die Anzeige gelesen würde.

Soll so etwas grundsätzlich nicht vorkommen, so ist am Universalzähler ein zusätzlicher Schalter vorzusehen, der entweder ein anderes Mikroprogramm einschaltet, dessen letzter Befehl aus einem Sprungbefehl besteht, der auf sich selbst weist, oder der Schalter verändert das Mikroprogramm entsprechend. Dies ist z. B. bei Verwendung der diskreten Diodenmatrix durch Einfügen der Dioden auf den Speicherplatz FH mit Hilfe eines Mehrfachschalters möglich. Die Dioden stellen dabei den Befehl springe nach F dar. Der Code hierfür lautet 4EH.

9.1.2 Universalzähler und periphere Anzeige

Zur Auswertung der Meßergebnisse ist es wünschenswert, alle Messungen zur Verfügung zu haben.

Abb. 9.1.2-1...3 zeigen verschiedene Möglichkeiten, die Daten zu erfassen. Die erste Möglichkeit besteht darin, das Datensichtgerät als Peripherie anzuschließen. Es ist so möglich,

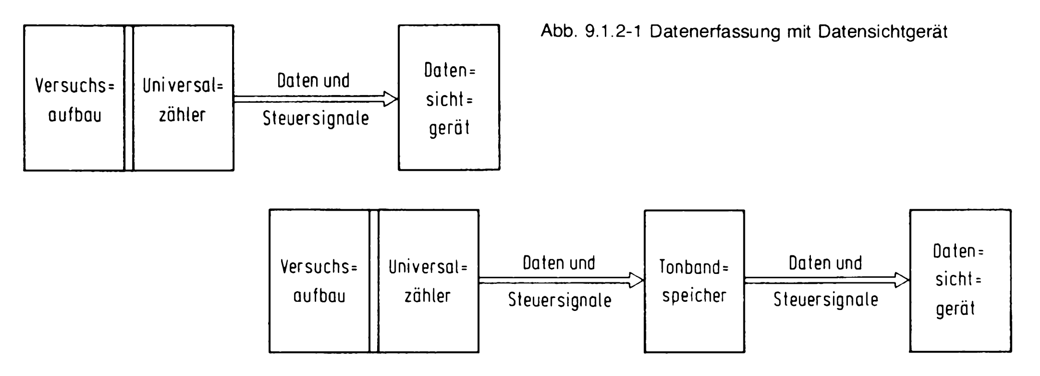


Abb. 9.1.2-1 Datenerfassung mit Datensichtgerät

Abb. 9.1.2-2 Datenerfassung mit Datensichtgerät und Tonbandspeicher

Abb. 9.1.2-3 Drucker als Peripheriegerät

Abb. 9.1.3-1 Datenerfassungssystem mit Mikrocomputer

anhand der Tabelle, die auf dem Datensichtgerät geschrieben wird, die Daten in Ruhe weiterzuverarbeiten. Die zweite Möglichkeit liegt darin, einen Tonbandspeicher zwischenzuschalten. Die Daten können dann dauerhaft festgehalten und später ausgewertet werden. Auch kann eine größere Datenmenge festgehalten werden, als sie auf dem Datensichtgerät dargestellt werden kann. Außerdem kann ein Drucker eingesetzt werden. Er bietet zusätzlich den Vorteil, die gesamte Datenmenge sichtbar festzuhalten.

9.1.3 Universalzähler, Datensichtgerät, Mikrocomputer

Abb. 9.1.3-1 zeigt diese Zusammenschaltung, die ein komplettes Datenerfassungssystem ergibt. Steht ein größerer Speicherplatz zur Verfügung, so kann eventuell die Rechenarbeit auch von dem Mikrocomputer bewältigt werden. Es sind dies die Mittelwertbildung und die anschließende Errechnung der Konstante g

mit der gegebenen Formel. Ein Flußdiagramm für eine solche Berechnung zeigt die *Abb. 9.1.3-2*. Es ginge zu weit, hier ein komplettes Assemblerprogramm zu entwickeln, da dies wahrscheinlich die in dem SDK 80 Kit vorhandene Speicherkapazität überschreiten würde. Es ist nämlich eine umfangreiche Arithmetik, vielleicht sogar mit Gleitpunkt, nötig, um diese Aufgabe zu bewältigen. Daher wurde ein Programm für einen wissenschaftlichen, programmierbaren Rechner, den SR 52 entwikkelt. Bei Auswerten der Daten leistet er ebenfalls gute Dienste, nur daß es nicht möglich ist, die Meßwerte direkt einzulesen. Die Ausgabe über einen Drucker ist allerdings möglich.

Abb. 9.1.3-3 zeigt ein genaueres Flußdiagramm zur Lösung der gestellten Aufgabe. Das Programm besteht aus zwei Teilen. Der erste Teil hat die Aufgabe, die Daten für die Mittelwertberechnung bereitzustellen. Der zweite Teil berechnet den Mittelwert und die Gravitationskonstante g. Der erste Teil funktioniert im Detail folgendermaßen: Nach Sprung auf die Marke A (label A) erfolgt ein Löschen und somit ein Rücksetzen der Speicherinhalte. Anschließend wird der Wert für $T/2$ manuell eingegeben. Es wird dann dieser eingegebene Wert in den Speicher mit dem

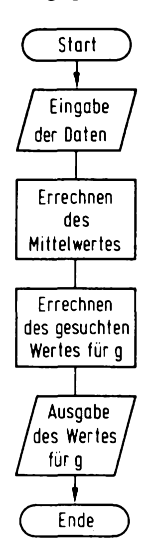


Abb. 9.1.3-2 Flußdiagramm für den allgemeinen Rechengang bei diesem Versuch

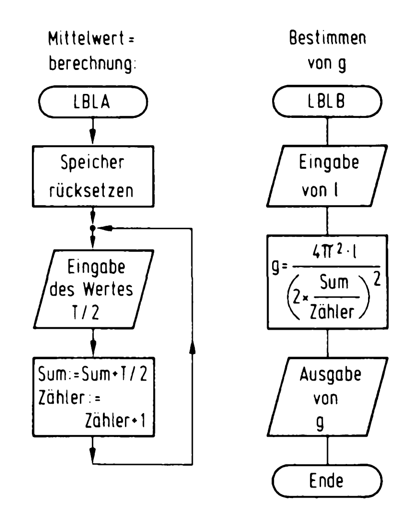


Abb. 9.1.3-3 Berechnung mit Hilfe des SR 52

Namen „SUM" zu dem Inhalt dieses Speichers addiert. Ein Speicher mit dem Namen „Zähler" wird um eins erhöht. Anschließend erfolgt ein Rücksprung zu dem Teil des Programms, an dem ein neuer Wert für $T/2$ eingegeben werden kann.

Wird zur Marke B (label B) gesprungen, so erfolgt die Berechnung der Gravitationskonstante und die Mittelwertberechnung. Es kann zuvor noch der Wert für die Länge des Pendels eingegeben werden. Die Marken werden angesprungen, wenn vom Benutzer die auf dem Rechner vorhandene Taste mit der gleichen Bezeichnung (also A oder B) betätigt wird. *Tabelle 9.1.3-1* zeigt das Programm für den SR52, das anhand des Flußdiagramms angefertigt wurde. *Tabelle 9.1.3-2* zeigt den Rest des Programms, das zur eigentlichen Berechnung der Gravitationskonstante g dient.

Noch ein paar Bemerkungen zum SR 52. Er besitzt einen Programmspeicher, der 224 Programmschritte speichern kann. Der SR 52 besitzt 20 adressierbare Datenspeicher, wissenschaftliche Funktionen, bedingte und unbedingte Verzweigungen, indirekte Adressierbarkeit von Daten und Programmspeicher, Unterprogrammsprungbefehle etc. Ferner ist

Tabelle 9.1.3-1

Schrift	Code	Befehl	Bemerkung
000	46	LBL	
001	11	A	
002	47	CMs	Löschen der Register
003	46	LBL	Marke für Schleife
004	87	1’	
005	81	HLT	Warten für Eingabe
006	44	SUM	Inhalt von
007	00	0	Register 01 um
008	01	1	Eingabe erhöhen
009	01	1	
010	44	SUM	Inhalt von Register
011	00	0	02 um 1 erhöhen
012	02	2	
013	43	RCL	Anzeigen von
014	00	0	Zählerinhalt
015	02	2	bzw. Anzahl der
016	41	GTO	Eingaben, Rücksprung
017	87	1’	für neue Eingabe

Tabelle 9.1.3-2

Schritt	Code	Befehl	Bemerkung
018	46	LBL	
019	12	B	
020	25	CLR	
021	81	HLT	Eingabe von l
022	65	*	
023	04	4	
024	65	*	
025	59	π	
026	40	X^2	
027	55	÷	
028	53	(	
029	43	RCL	
030	00	0	
031	01	1	
032	55	÷	
033	43	RCL	
034	00	0	
035	02	2	
036	65	*	
037	02	2	
038	54	)	
039	40	X^2	
040	95	=	
041	81	HLT	Anzeige des Ergebnisses

es möglich, das Programm auf einer Magnetkarte festzuhalten. Mit einem Hilfsprogramm ist es sogar möglich, den Inhalt des Datenspeichers aufzuzeichnen.

9.2 Elektronisches Labyrinth

Hier soll nun anhand eines weiteren interessanten Beispiels das Arbeiten mit Software gezeigt werden. Aufgabe des ersten Abschnitts ist es, mit Hilfe eines Programms die Möglichkeit zu geben, ein Labyrinth, wie es *Abb. 9.2-1* zeigt, auf das Datensichtgerät bedienerfreundlich aufzuzeichnen.

9.2.1 Eingabeprogramm für Datensichtgerät

Dieses Programm soll es dem Benutzer ermöglichen, durch Betätigen von Tasten der alphanumerischen Tastatur den Cursor zu positionieren. Dabei soll der Taste „W“ die Bedeutung „Cursor um eins nach oben“, der Taste „S“ die Bedeutung „Cursor um eins nach rechts“, der Taste „A“ die Bedeutung „Cursor um eins nach links“ und der Taste „Z“ die Bedeutung „Cursor um eins nach unten“ gegeben werden. *Abb. 9.2.1-1* zeigt die Anordnung der gewählten Tasten auf der Tastatur.

Auf dieser Abbildung sind auch noch zwei andere Tasten dargestellt. Die Taste „X“ und die Taste „space“. Das Programm soll diese beiden Tasten auch noch umdefinieren, und zwar soll bei Betätigen der Taste „X“ das Zeichen gelöscht werden, das sich auf der Position des Cursors befindet, anschließend soll der Cursor wieder auf der Stelle sein, auf der er vorher war. Die Taste „space“ soll das Zeichen „*“ auf die Stelle des Bildschirms eingeben, auf der sich der Cursor befindet. Dann soll der Cursor wieder auf diese Stelle zurückkehren.

Durch diese Maßnahmen soll ein bequemes Eingeben des Labyrinths ermöglicht werden.

Programmbeschreibung

Zunächst noch etwas allgemeines zum Planen von Programmen: Es ist vorteilhaft, ein größe-

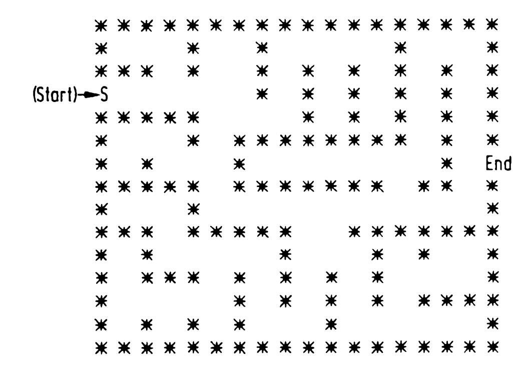


Abb. 9.2-1 Darstellung des Labyrinths

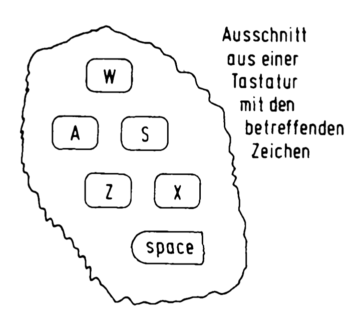


Abb. 9.2.1-1 zu definierende Tasten

res Programm in möglichst viele Unterprogramme zu zerlegen, es sei denn, es kommt auf eine hohe Ausführungsgeschwindigkeit an.

Dadurch ergibt sich neben einer größeren Übersichtlichkeit der zusätzliche Vorteil, daß die Möglichkeit besteht, die Programme stufenweise auszutesten und zum Laufen zu bringen. Auch das Programm „Labyrinth" ist in viele solcher Unterprogramme eingeteilt.

In diesem Abschnitt wird der erste Programmteil des Labyrinthprogramms gebracht, der für sich allein schon ein funktionstüchtiges Programm darstellt. Dieser Programmteil besteht aus einem Hauptprogramm mit dem Namen MAIN 1 und einigen Unterprogrammen. Am besten wird mit der Besprechung der Unterprogramme begonnen.

Abb. 9.2.1-2 zeigt das Unterprogramm mit dem Namen UP. Dieses Unterprogramm hat die Aufgabe, zwei Registerinhalte zu definieren. Der Inhalt des Registers C wird mit dem Code für „up" besetzt. Der Code für diesen Befehl ist in der Tabelle 4.2.9-1 zu finden. Später soll dieser Wert über den seriellen Peripherieausgang des SDK 80 Kit ausgegeben werden. Das Monitorprogramm enthält dazu ein Unterprogramm, das die Ausgabesteuerung übernimmt. Um das Programm zu verwenden, muß das auszugebende Zeichen in das Register C des Mikroprozessors eingespeichert werden. Dann kann das Unterprogramm mit dem Namen CO aufgerufen werden. Das Unterprogramm UP besetzt noch das Register D mit dem Wert 02H (H bedeutet Hexadezimal). Diese Zuweisung ist nötig, um später dem Programmteil MOUSE zu ermöglichen, festzustellen, wo sich eine Mauer befindet. Doch darüber später.

Abb. 9.2.1-3 zeigt das Flußdiagramm für das Unterprogramm RIGHT, *Abb. 9.2.1-4* das Flußdiagramm für das Unterprogramm LEFT und *Abb. 9.2.1-5* das Flußdiagramm für das Unterprogramm DOWN. In diesen Unterpro-

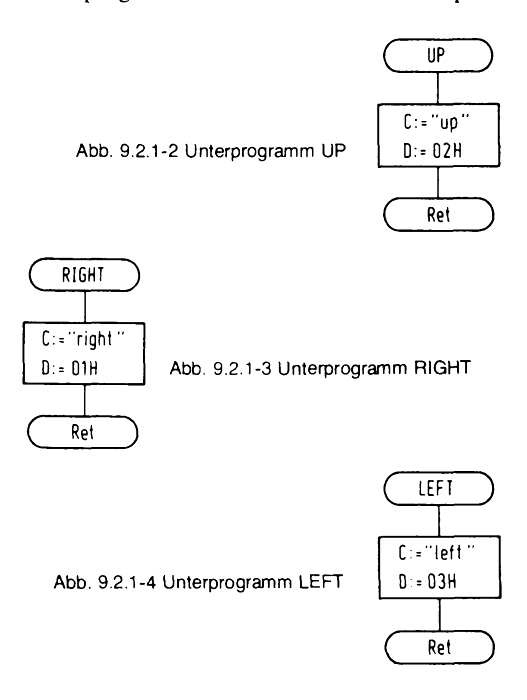


Abb. 9.2.1-2 Unterprogramm UP

Abb. 9.2.1-3 Unterprogramm RIGHT

Abb. 9.2.1-4 Unterprogramm LEFT

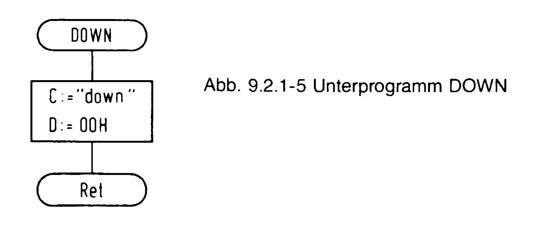


Abb. 9.2.1-5 Unterprogramm DOWN

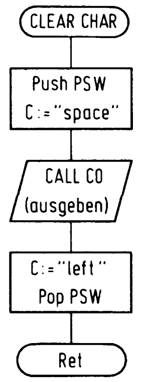


Abb. 9.2.1-6 Unterprogramm CLEAR CHAR

grammen wird entsprechend dem Unterprogramm UP verfahren. Es werden die Register C und D mit den Werten besetzt, die für das Unterprogramm spezifisch sind.

Abb. 9.2.1-6 zeigt das Flußdiagramm für das Unterprogramm mit dem Namen CLEAR CHAR. Dieses Flußdiagramm bedarf nun wieder einer etwas genaueren Erklärung. Das Unterprogramm hat die Aufgabe, das Zeichen, auf dem sich der Cursor befindet, zu löschen und danach den Cursor wieder an seine ursprüngliche Stelle zurückzusetzen.

Das Löschen des Zeichens geschieht mit dem Zeichen „space“, also, indem die betreffende Speicherzelle des Datensichtgeräts mit dem Leerzeichen überschrieben wird. Da sich der Cursor nach Ausführung dieses Befehls um eins nach rechts bewegt hat, ist es dann notwendig, den Befehl „Rückwärtsschritt“ auszuführen (Befehl „left“). Es genügt bei diesem Unterprogramm nicht mehr, das Register C mit einem Zeichen zu besetzen, sondern das Register muß sozusagen zweimal besetzt werden. Dazwischen muß ein Wert ausgegeben werden.

Da das Unterprogramm mit dem Ausgabebefehl des Monitors das Register A (Akkumulator) verändern würde, der Inhalt des Registers A aber noch benötigt wird, ist es notwendig, diesen Wert gesondert festzuhalten. Dies geschieht am einfachsten mit dem Befehl PUSH PSW. Er bewirkt ein Abspeichern des Registers A und ein Abspeichern des Programmstatuswortes (Zustandsbit). Dabei wird diese Information auf einen Stack abgespeichert, dessen Adresse durch den Inhalt des Registerpaars SP (stackpointer) bestimmt ist. Dieses Registerpaar wird automatisch vom Monitorprogramm auf einen definierten Wert gesetzt, der den Stack an den oberen Teil des Speicherbereichs mit den Adressen 1300H...13FFH setzt.

Nach dem Befehl PUSH PSW ist auf dem Flußdiagramm der Befehl, der das Register C mit dem Code für „space“ belegt, eingezeichnet. Anschließend erfolgt die Ausgabe dieses Codes an das Peripheriegerät (hier das Datensichtgerät) mit Hilfe des Befehls CALL CO. Danach erfolgt die Zuweisung des Codes für „left“ an das Register C. Mit dem Befehl POP PSW wird das zuvor abgespeicherte Programmstatuswort und der Inhalt des Akkumulators wieder zurückgeschafft, so daß wie gewünscht der Akkumulator wieder den alten Wert enthält. Es erfolgt dann der Rücksprung zum aufrufenden Programm.

Abb. 9.2.1-7 zeigt das Flußdiagramm für das Unterprogramm „*“. Es hat die Aufgabe, das Zeichen „*“ an die Stelle des Bildschirms zu schreiben, auf der der Cursor steht. Anschließend wird der Cursor, der ja eine Position weitergewandert ist, wieder auf diese Ausgangsposition zurückbefördert. Das Programm ähnelt dem vorhergehenden, nur daß nicht das Zeichen „space“, sondern das Zeichen „*“ zunächst in das Register C geladen wird.

Abb. 9.2.1-8 zeigt das letzte Unterprogramm für diesen Programmteil, das den Namen „E“ trägt. Es hat lediglich die Aufgabe,

den im Akkumulator enthaltenen Wert in das Register C zu befördern, um so eine Ausgabe des Zeichens zu ermöglichen.

Abb. 9.2.1-9 zeigt nun den Programmteil MAIN 1, genauer gesagt, den ersten Teil davon. Das Programm beginnt mit der Zuweisung des Wertes 00H an das Register E. Dieses Register wird in einem späteren Programmteil noch gebraucht. Anschließend erfolgt ein Eingabebefehl mit der Bezeichnung CALL CI. Dieser Befehl ruft ein Unterprogramm auf, das im Monitorprogramm enthalten ist. Es hat die Aufgabe, solange zu warten, bis ein Zeichen über den seriellen Eingang des Peripherie-ICs des Mikrocomputers eingegeben wurde. Dann hat das Unterprogramm CI noch die Aufgabe, dieses Zeichen in das Register A zu befördern. Nach Ausführung dieses Unterprogramms erfolgt ein bedingter Sprung. Der Inhalt des Registers A, in das soeben ein Zeichen eingegeben wurde, stellt dabei die Entscheidungsgrundlage dar. Der Inhalt wird zunächst mit dem Zeichen „W“ verglichen. Ist es das Zeichen „W“, so erfolgt die Ausführung des Unterprogramms UP. Als nächstes wird das eingegebene Zeichen mit dem Zeichen „S“ ver-

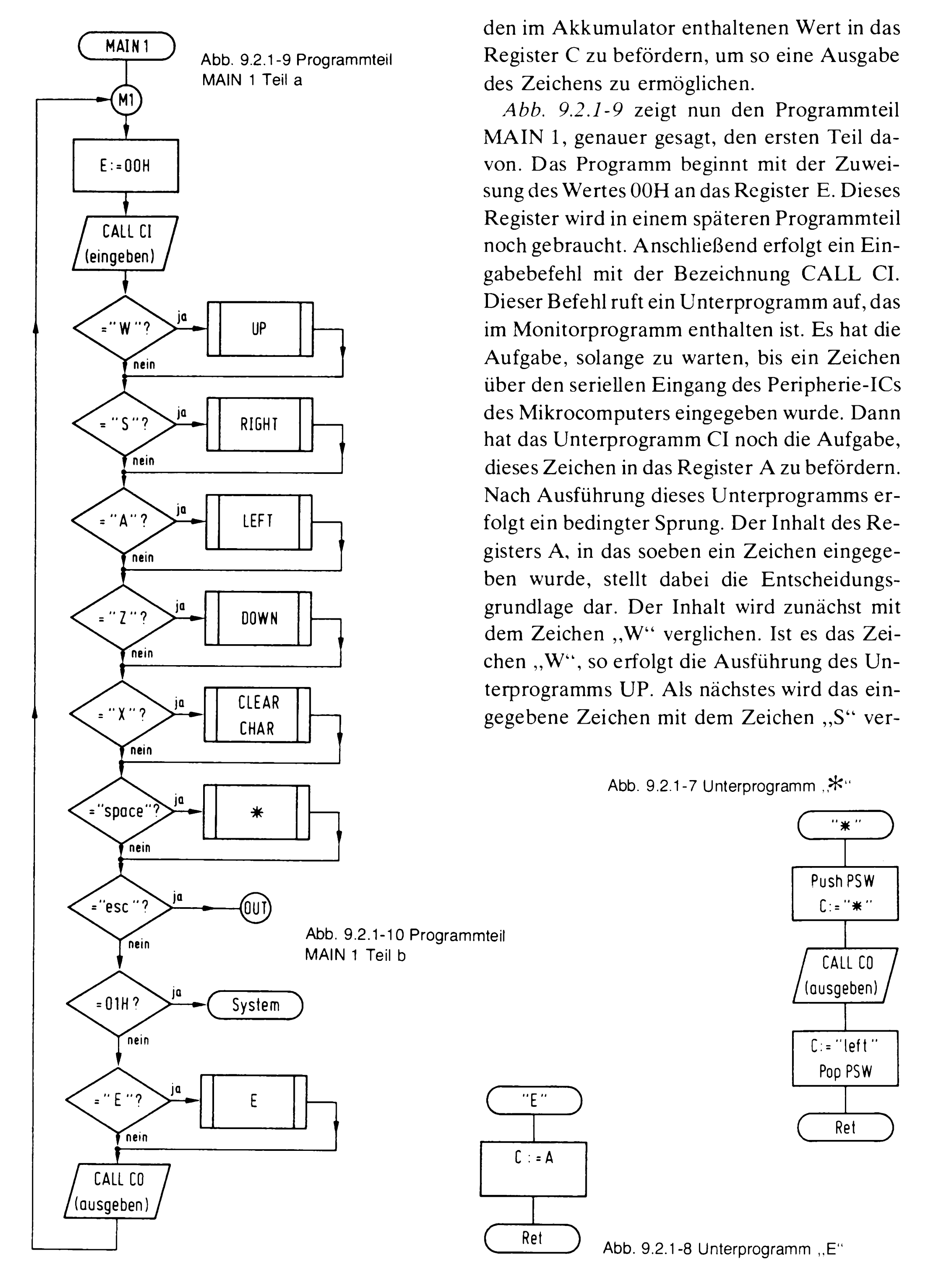


Abb. 9.2.1-9 Programmteil MAIN 1 Teil a

Abb. 9.2.1-10 Programmteil MAIN 1 Teil b

Abb. 9.2.1-7 Unterprogramm „*“

Abb. 9.2.1-8 Unterprogramm „E“

glichen. Ist es das Zeichen „S“, so erfolgt die Ausführung des Unterprogramms RIGHT. Entsprechend wird mit den anderen Zeichen verfahren.

Der Vergleich erfolgt z. B. mit dem Befehl CPI. Dieser Befehl bewirkt einen Vergleich des Inhalts des Akkumulators mit dem der Instruktion folgenden Byte dadurch, daß nach einer Subtraktion die Zustandsbits entsprechend dem Ergebnis gesetzt werden. Der Inhalt des Akkumulators wird dabei nicht verändert. Mit dem Befehl CZ (call if zero) kann das hier interessierende Nullbit abgefragt werden. Wenn das Nullbit gesetzt wurde, der Inhalt des Akkumulators also gleich dem der Instruktion folgenden Byte ist, so erfolgt der Aufruf des Unterprogramms, dessen Adresse in den beiden folgenden Bytes des Befehls CZ gegeben ist.

Abb. 9.2.1-10 zeigt den Rest des Programms MAIN 1. Es erfolgt nun ein Vergleich mit dem Zeichen „esc“. Wenn es sich um das Zeichen „esc“ handelt, das eingegeben wurde, so erfolgt ein Sprung zu einem Programm mit dem Namen OUT. Dieses Programm wird in dem nächsten Abschnitt besprochen. Wenn es sich nicht um das Zeichen „esc“ gehandelt hat, so erfolgt ein Vergleich mit dem Code 01H. Ist der Code 01H mit dem Code des eingegebenen Zeichens identisch, so erfolgt ein Sprung in das Monitorprogramm (Einsprungadresse ist 0008H).

Es folgt im anderen Falle der Vergleich mit dem Zeichen „E“. Handelt es sich um das Zeichen „E“, so wird das Unterprogramm mit der gleichnamigen Bezeichnung aufgerufen. Dadurch wird ermöglicht, das Zeichen „E“ auszugeben. Dieses Zeichen wird später benötigt, um das Ende des Labyrinths automatisch zu erkennen.

Als nächster Befehl folgt CALL CO. Damit wird das in Register C vorhandene Zeichen ausgegeben. Anschließend erfolgt ein Rücksprung zur Marke „M1“. Dadurch wird das Programm MAIN 1 wiederholt.

9.2.2 Labyrinthprogramm 2. Teil

Während das erste Programm Bedienungen vereinfachte, und notfalls auch durch die Bedientasten ersetzt werden könnte, ist dieses Programm nicht mehr durch Bedientasten des Datensichtgerätes ersetzbar. Es verwendet die Unterprogramme des ersten Programms und besitzt weitere eigene Unterprogramme und ein Hauptprogramm mit dem Namen OUT.

Es wird zunächst das Unterprogramm READ nach *Abb. 9.2.2-1* besprochen. Es hat die Aufgabe, ein Zeichen von dem Bildschirm zu lesen, das durch die Position des Cursors bestimmt ist, und dabei die Position des Cursors zu erhalten. Ferner muß das Unterprogramm noch ein Zeichen ausgeben, das sich im Register C befindet. Das Programm beginnt also mit dem Befehl CALL CO. Damit ist das Zeichen des Registers C ausgegeben. Danach wird das Register mit dem Code für den Befehl „read“ (0EH) belegt.

Der Code wird anschließend ausgegeben. Dadurch wird das Datensichtgerät in die Betriebsart „Lesen“ umgeschaltet. Die Eingabe des Zeichens in den Mikrocomputer erfolgt nun dadurch, daß das Zeichen „?“ ausgegeben wird! Anschließend erfolgt die Eingabe des schon im UART des Mikrocomputersystems vorhandenen Zeichens in das Register A durch den Befehl CALL CI. Anschließend wird der Inhalt des Registers A in das Register B kopiert. Dadurch wird erreicht, daß dieser Wert nicht verlorengeht. Das Register C wird nun mit dem Code für den Befehl „write“ (0FH) besetzt und ausgegeben. Es folgt dann die Ausgabe des Befehls „Rückschritt“ (left oder BS), um den Cursor wieder auf die „alte“ Stelle zu positionieren. Zuletzt wird der Inhalt des Registers B wieder in das Register A kopiert.

Es folgt dann der obligatorische Rücksprungbefehl. *Abb. 9.2.2-2* zeigt das Flußdiagramm für das Programm MUP.

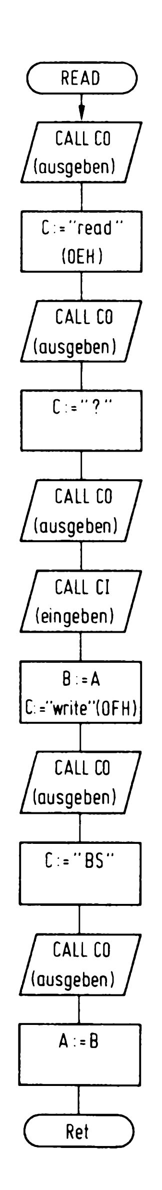


Abb. 9.2.2-1 Unterprogramm READ bzw. Programmteil READ

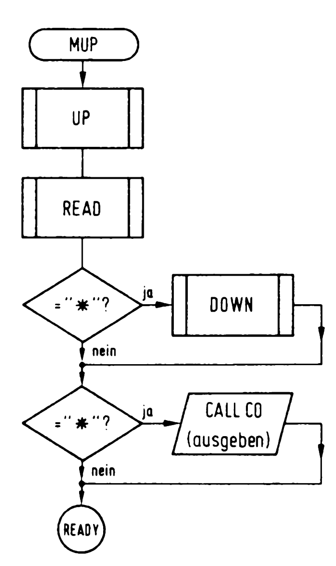


Abb. 9.2.2-2 Unterprogramm MUP bzw. Programmteil MUP

Gleich nach Start dieses Programms erfolgt der Aufruf des Unterprogramms UP, danach der Aufruf des Unterprogramms READ. Somit wird der Cursor um eine Position nach oben gerückt und das dort befindliche Zeichen vom Bildschirm in das Register A eingelesen. Wenn es sich bei dem eingelesenen Zeichen um das Zeichen „*" handelt, so erfolgt der Aufruf des Unterprogramms DOWN. Anschließend erfolgt der Aufruf des Unterprogramms CO, falls das Zeichen „*" vorhanden war. Eine Mauer im Labyrinth wird ja mit dem Zeichen „*" dargestellt. Durch dieses Unterprogramm wird erreicht, daß eine Bewegung in Richtung „Nach oben" nur dann erfolgen kann, wenn keine Mauer vorhanden ist. Ist eine Mauer vorhanden, so wird die Bewegung „Nach oben" zwar zunächst doch durchgeführt, doch dann wird die Bewegung wieder rückgängig gemacht.

Ähnlich wird mit den anderen Richtungen verfahren. *Abb. 9.2.2-3* zeigt das Unterprogramm für die Richtung RIGHT mit dem Namen MRIGHT. *Abb. 9.2.2-4* zeigt das Unterprogramm für die Richtung LEFT mit dem Namen MLEFT, und *Abb. 9.2.2-5* zeigt das Unterprogramm für die Richtung DOWN mit dem Namen MDOWN. Nach Beendigung dieser Programme erfolgt stets ein Sprung zu einem Programmteil mit dem Namen READY. Diesen Programmteil zeigt *Abb. 9.2.2-6*.

Der erste Befehl dieses Programmteils weist dem Register A den Wert des Registers B zu. Es erfolgt dann die Überprüfung auf das Zei-

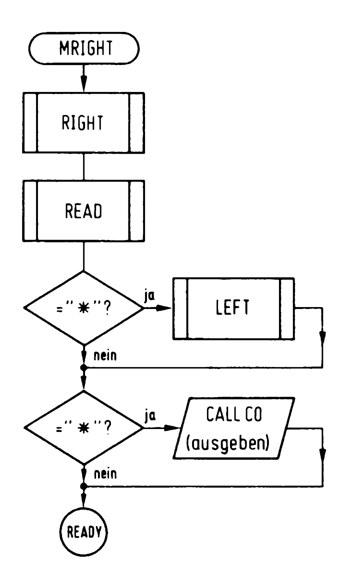


Abb. 9.2.2-3 Unterprogramm MRIGHT bzw. Programmteil MRIGHT

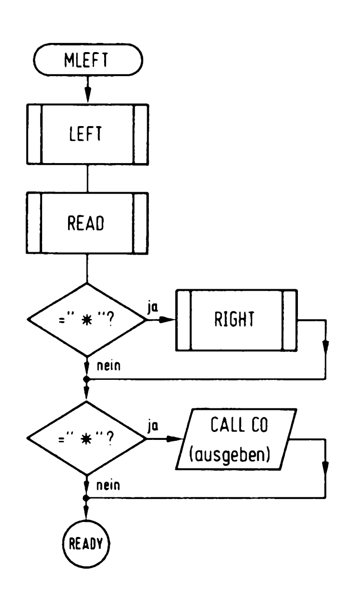


Abb. 9.2.2-4 Unterprogramm MLEFT bzw. Programmteil MLEFT

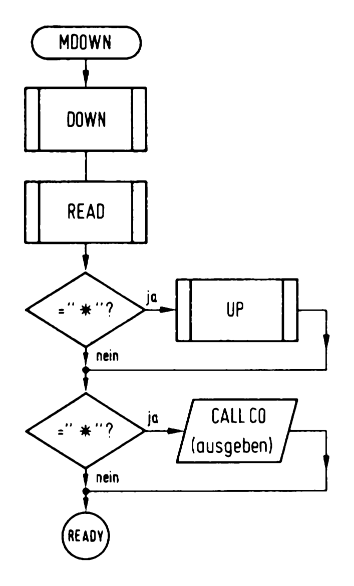


Abb. 9.2.2-5 Unterprogramm MDOWN bzw. Programmteil MDOWN

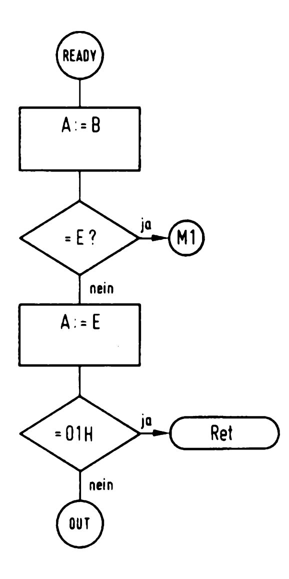


Abb. 9.2.2-6 Unterprogramm READY bzw. Programmteil READY

chen E. Handelt es sich um das Zeichen E, so war an der Stelle der Abfrage das Zeichen E auf dem Bildschirm vorhanden, und somit wurde das Ende des Labyrinths erreicht. Es erfolgt dann ein Sprung zum Programmteil M1 und das Programm MAIN 1 wird wieder durchgeführt. Ist es nicht das Zeichen E, so wird im Programmteil READY fortgefahren. Dem Register A wird der Wert des Registers E zugewiesen. Ist der Inhalt des Registers A dann identisch mit dem Wert 01H, so erfolgt ein Rücksprung, andernfalls ein Sprung zu dem Programmteil OUT. Es ist möglich, den Programmteil wahlweise als Unterprogramm oder als normales Programm zu verwenden. In Abhängigkeit des Inhalts des Registers E erfolgt die Auswahl.

Dabei ist folgendes zu beachten. Falls dieses Programmteil als Unterprogramm verwendet wird, wie dies später von dem Programmteil MOUSE getan wird, und ein Sprung nach M1 erfolgt, wodurch das Ende erreicht wurde, so erfolgt kein Rücksprung mehr. Der Stackpointer erreicht dadurch seinen alten Wert nicht mehr und der Stack selbst wird weiter nach „unten" verlegt. Erfolgt dies sehr häufig, so kann es passieren, daß der Stack mit dem Teil des Speichers zusammenfällt, in dem sich das Programm befindet. Wird das Programm MOUSE mehrmals aufgerufen, und wird mehrmals das Ende des Labyrinths von der „Maus" gefunden, ist es also nötig, den Reset des Systems auszulösen. Dies geschieht dadurch, daß die Tasten „CTRL !" betätigt werden, nachdem das Ende des Labyrinths von der „Maus" gefunden wurde. Dadurch erfolgt ein Sprung ins Monitorprogramm und somit eine neue Besetzung des Stackpointers.

Das Programm OUT, das *Abb. 9.2.2-7* zeigt, hat die Aufgabe, die vorher besprochenen Programmteile entsprechend der Eingabe des Benutzers aufzurufen. Dazu wird als erstes mit dem Befehl CALL CI auf eine Eingabe vom Benutzer gewartet. Ist diese erfolgt, so wird das eingegebene Zeichen zunächst mit

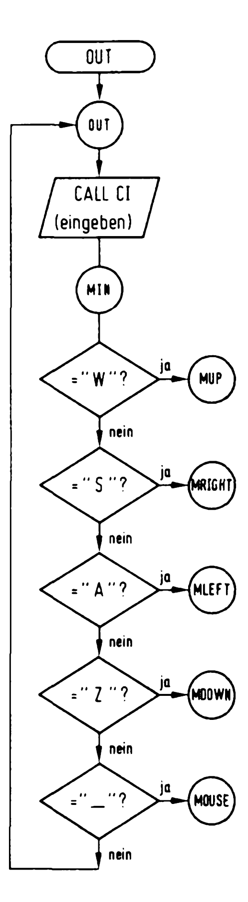


Abb. 9.2.2-7 Programmteil OUT

dem Zeichen „W" verglichen. Handelt es sich um das Zeichen „W", so erfolgt ein Sprung zum Programmteil MUP. Andernfalls wird das eingegebene Zeichen mit dem Zeichen „S" verglichen. Im Falle der Übereinstimmung wird ein Sprung zum Programmteil MRIGHT ausgeführt. Wenn nicht, erfolgt der Vergleich mit dem Zeichen „A" und im Falle der Übereinstimmung der Sprung zum Programmteil MLEFT. Wenn nicht, erfolgt weiter der Vergleich mit dem Zeichen „Z" und im Falle der Übereinstimmung der Sprung zum Programmteil MDOWN. Wenn nicht, erfolgt noch der Vergleich mit dem Zeichen „_" und falls dieses Zeichen eingegeben wurde, erfolgt ein Sprung

zu dem Programm MOUSE, das in dem nächsten Kapitel genauer besprochen wird. Wenn nicht, erfolgt ein Sprung zur Marke OUT.

Benutzerhinweise
Wie arbeitet man nun mit den bisher besprochenen Programmen? Als erstes erfolgt ein Start des Programms MAIN 1 (siehe auch nächstes Kapitel). Damit ist es möglich, ein beliebiges Labyrinth einzugeben. Danach erfolgt die Betätigung der Taste „esc". Es folgt nun das manuell gesteuerte Bewegen innerhalb des Labyrinths. Dazu stehen die schon erklärten Tasten W, S, A, Z zur Verfügung. Wenn das Ende des Labyrinths erreicht wurde, das auf dem Bildschirm mit E bezeichnet ist, so erfolgt automatisch wieder der Eingabe-Modus, also es ist dann möglich, entweder das Programm durch Betätigen der Tasten „CTRL !" zu verlassen, oder es kann das Labyrinth neu eingegeben oder verbessert werden. Wird nach Betätigen der Taste „esc" die Taste „–" betätigt, so erfolgt eine Übergabe an das Programm MOUSE.

9.3 Labyrinth, Kybernetisches Modell für Verhalten einer Maus

Hier soll das schon erwähnte Programm MOUSE erklärt werden.

Das Labyrinth, wie es zum Beispiel Abb. 9.2-1 zeigt, muß den Ausgang auf der Außenseite haben, damit die Maus auch den Ausgang mit Sicherheit finden kann.

Die „Maus" besitzt eine Suchtaktik. Bei Durchlaufen des Labyrinths hält sich die Maus immer an der rechten Mauer. Dadurch ist es der Maus möglich, immer den Ausgang zu finden.

Das Programm MOUSE beginnt mit der Zuweisung des Wertes 01H an das Register E. Dadurch wird erreicht, daß bei Aufruf der Programmteile MUP, MDOWN, MLEFT, MRIGHT diese als Unterprogramme verwendet werden.

Die Register B und C werden mit dem Wert 00H besetzt. Das Registerpaar HL erhält die Adresse einer Liste zugewiesen, die noch erklärt wird.

Es folgt nun das Sichern des Inhalts des Registerpaars HL durch den Befehl PUSH H. Anschließend wird der Inhalt des Registerpaars BC (in dem Fall des ersten Durchlaufs 00H) zum Inhalt des Registerpaars HL dazugezählt. Das Ergebnis der Addition wird in dem Registerpaar HL abgespeichert. Anschließend wird mit dem Befehl „A:=M" oder, entsprechend dem Befehlssatz des 8080: MOV A, M, der Inhalt des durch die Adresse im Registerpaar HL bestimmten Speicherplatzes in das Register A geladen. Nun erfolgt die Abspeicherung des Inhalts des Registerpaars BC durch den Befehl PUSH B. Dann erfolgt der Aufruf des Programms OUT bei der Marke MIN beginnend.

Das Programm OUT läuft dann ab, wie schon beschrieben, nur daß erstens die Eingabe, die beim normalen Aufruf dieses Programms durch den Benutzer erfolgt, hierbei nicht durchgeführt wird, sondern durch den Befehl A:=M ersetzt wurde, und daß zweitens ein Rücksprung in das Programm MOUSE erfolgt.

Nach Ausführung des Unterprogramms MIN wird die Rückspeicherung des Registerpaars BC und die Rückspeicherung des Registerpaars HL vorgenommen. Anschließend wird der Inhalt des Registers C in das Register A kopiert. Ist dieser Inhalt des Registers A mit dem Inhalt des Registers D identisch, so erfolgt ein Sprung zur Marke 1. Man erinnere sich: Bei Ausführung eines der vier am Anfang des Kapitels 9.2.2 besprochenen Unterprogramme UP, RIGHT, LEFT, DOWN wird das Register D mit Werten belegt. Es ist nämlich jetzt möglich, durch Vergleich von Register C mit Register D festzustellen, ob eine Mauer in der Richtung vorhan-

den war, die gerade ausprobiert wurde. Es ist also möglich, festzustellen, ob der Zug in der durch die Liste bestimmten Richtung tatsächlich durchgeführt wurde oder nicht. Wenn der Inhalt des Registers D gleich dem Inhalt des Registers C ist, dann wurde der Zug wirklich ausgeführt.

Dann erfolgt die Erniedrigung des Inhalts des Registers A um eins. Andernfalls, wenn der Zug nicht ausgeführt wurde, erfolgt die Erhöhung des Inhalts des Registers A um eins. Nun wird in beiden Fällen die UND-Verknüpfung des Inhalts des Registers A mit dem Wert 03H durchgeführt. Dadurch wird erreicht, daß der Wert des Registers A nicht größer als 03H wird. Dann geschieht der Sprung zu der Marke LOOP und somit eine Wiederholung eines Teils des Programms MOUSE.

Es folgt nun die Erklärung der Bedeutung und Funktion der Liste nach *Tabelle 9.3-1*. Auf dieser Tabelle ist die Adresse des Speicherplatzes dargestellt und der Code der Information, die auf diesem Speicherplatz zu finden ist.

Dabei steht auf dem ersten Speicherplatz mit der Adresse 00H der Code für das Zeichen „Z", dessen Bedeutung mit DOWN definiert wurde, auf dem Speicherplatz mit der Adresse 01H der Code für das Zeichen „S" und somit die Bedeutung RIGHT. Ebenso folgen UP und LEFT. Die Liste wird nun irgendwo im Speicher untergebracht. Bei Aufruf des Programms MOUSE erhält das Registerpaar HL die Adresse des Anfangs dieser Liste zugewiesen. Dem Registerpaar BC wird der Wert 0000H zugewiesen. Dadurch wird die erste Stelle der Liste bei der ersten Ausführung des Befehls A:=M in das Register A kopiert. Es erfolgt somit bei Ausführung des Unterprogramms MIN ein Versuch der Maus, einen ersten Schritt „Nach unten" zu unternehmen. Anhand *Abb. 9.3-2* wird am besten der weitere Verlauf verfolgt.

Die Maus befindet sich vor der Ausführung einer Bewegung auf einer Stelle, die mit S bezeichnet wurde. Die Maus versucht nun zu-

Tabelle 9.3-1

Adresse	Code	Bedeutung
00	5A	Z (down)
01	53	S (right)
02	57	W (up)
03	41	A (left)

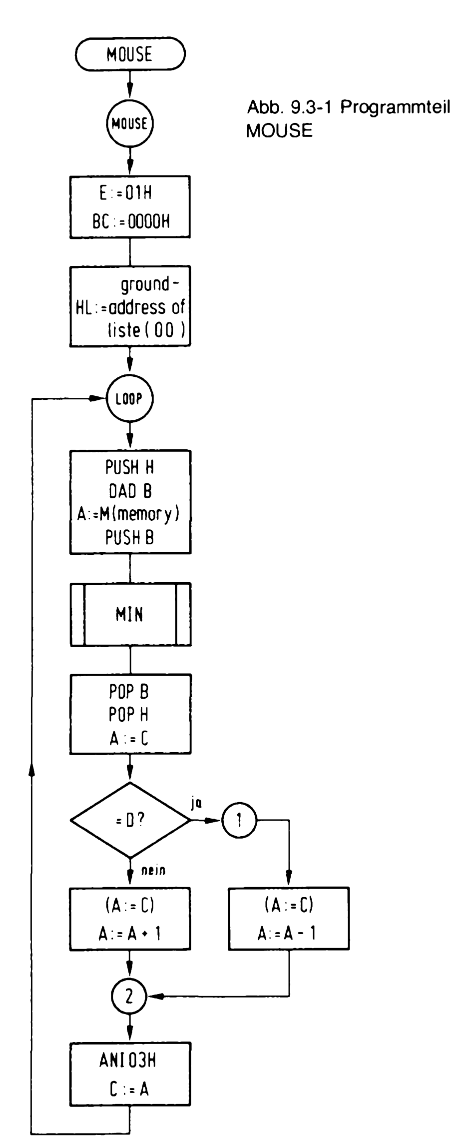


Abb. 9.3-1 Programmteil MOUSE

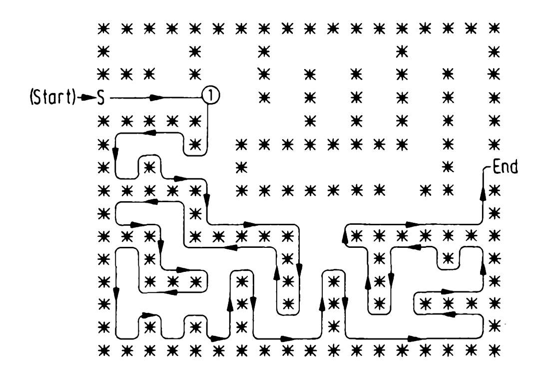


Abb. 9.3-2 Der Weg der „Maus“

nächst, einen Schritt nach unten. Dies ist aber nicht möglich, da sich dort eine Mauer befindet. Die Maus erkennt dies dadurch, daß bei dem Vergleich des Registers C mit dem Register D keine Übereinstimmung festgestellt werden kann. Das Register C enthält nämlich den Wert 00H und das Register D den Wert 02H, den es bei der Ausführung des Unterprogramms UP erhalten hatte. Nun wird, da keine Übereinstimmung vorhanden war, der Inhalt des Registers A, das den Wert des Registers C erhielt, um eins erhöht und nach der UND-Verknüpfung, die hier keine Veränderung des Inhalts mit sich bringt, wieder in das Register C befördert. Das Register enthält dann den Wert 01H. Es erfolgt bei Ausführung des Unterprogramms der Versuch der Maus, nach rechts zu gehen, da die Stelle der Liste mit der Adresse 01H den Code für die Bedeutung „rechts“ enthält. Der Zug nach rechts ist möglich, und die Maus erfährt dies durch Vergleich der Register C und D. Es folgt eine Erniedrigung des Inhalts des Registers C um den Wert eins. Dadurch wird beim nächsten Versuch der Maus als erstes eine Bewegung nach unten versucht. Da dies nicht möglich ist, versucht die Maus nun eine Bewegung nach rechts und gelangt dann wieder ein Stück weiter. Nun wird wieder eine Bewegung nach unten versucht, dann eine nach rechts usw., bis die Maus auf die Position 1 gelangt.

Dort ist eine Bewegung nach unten möglich, und der Inhalt des Registers A, das den Wert 00H von Register C erhalten hat, wird um eins erniedrigt. Das Register A enthält danach den Wert FFH. Durch die UND-Verknüpfung mit dem Wert 03H wird erreicht, daß das Register den Wert 03H enthält. Damit wird bei der nächsten Ausführung des Programmteils MIN ein Versuch der Maus gemacht, nach links zu gehen. Dies gelingt aber nicht. Es erfolgt ein weiterer Versuch nach unten, der gelingt. In dieser Weise geht es fort, bis die Maus das Ende des Labyrinths erreicht hat. In Abb. 9.3-2 ist der gesamte Suchlauf der Maus bis zum Erreichen des Endes eingezeichnet. Der Übersicht wegen wurde die Linie, die diesen Lauf darstellt, so eingezeichnet, daß an Stellen. an denen die Maus zum Beispiel aus einer Sackgasse wieder zurückläuft, die Hinlauf- und die Rücklauf-Linie getrennt erkennbar sind.

Möglichkeiten für den Benutzer, das Programm zu verbessern

Den Maschinencode zeigt *Abb. 9.3-3* für eine Start-Adresse 1000H. Das Programm kann dabei mit den Befehlen G 1032 CR gestartet werden. (Auf der Adresse 1032H liegt der Beginn des Programms MAIN 1.)

Es sind in diesem Programm einige Möglichkeiten vorhanden, das Programm zu verbessern, zu kürzen oder zu optimieren.

Beispiel: Adresse 1113H des dargestellten Maschinencodes. Auf diesem Platz befindet sich ein Befehl, der weggelassen werden kann. Ein anderes Problem, das man versuchen kann einmal selbst zu lösen, ist folgendes.

Wie schon erwähnt, kann es Schwierigkeiten geben, wenn das Programm MOUSE sehr oft aufgerufen wird, ohne daß das Monitorprogramm aufgerufen wurde. Man kann diesen Fehler auf einfache Art einmal selbst zu beheben versuchen. Als Hinweis sei gesagt, daß

```
1000 0E 0B 16 02        1090 15 10 CC FA
1004 C9 00 00 0E        1094 03 C3 C8 10
1008 09 16 01 C9        1098 CD 07 10 CD
100C 00 00 0E 08        109C 6A 10 FE 2A
1010 16 03 C9 00        10A0 CC 0E 10 CC
1014 00 0E 0A 16        10A4 FA 03 C3 C8
1018 00 C9 00 00        10A8 10 CD 0E 10
101C F5 0E 20 CD        10AC CD 6A 10 FE
1020 FA 03 0E 08        10B0 2A CC 07 10
1024 F1 C9 F5 0E        10B4 CC FA 03 C3
1028 2A CD FA 03        10B8 C8 10 CD 15
102C 0E 08 F1 C9        10BC 10 CD 6A 10
1030 4F C9 1E 00        10C0 FE 2A CC 00
1034 CD FD 03 FE        10C4 10 CC FA 03
1038 57 CC 00 10        10C8 78 FE 45 CA
103C FE 53 CC 07        10CC 32 10 7B FE
1040 10 FE 41 CC        10D0 01 C8 C3 D5
1044 0E 10 FE 5A        10D4 10 CD FD 03
1048 CC 15 10 FE        10D8 FE 57 CA 87
104C 58 CC 1C 10        10DC 10 FE 53 CA
1050 FE 20 CC 26        10E0 98 10 FE 41
1054 10 FE 1B CA        10E4 CA A9 10 FE
1058 D5 10 FE 01        10E8 5A CA BA 10
105C CC 08 00 FE        10EC FE 5F CA F8
1060 45 CC 30 10        10F0 10 C3 D5 10
1064 CD FA 03 C3        10F4 5A 53 57 41
1068 32 10 CD FA        10F8 1E 01 01 00
106C 03 0E 0E CD        10FC 00 21 F4 10
1070 FA 03 0E 3F        1100 E5 09 7E C5
1074 CD FA 03 CD        1104 CD D8 10 C1
1078 FD 03 47 0E        1108 E1 79 BA CA
107C 0F CD FA 03        110C 13 11 79 3C
1080 0E 08 CD FA        1110 C3 15 11 79
1084 03 78 C9 CD        1114 3D E6 03 4F
1088 00 10 CD 6A        1118 C3 00 11 00
108C 10 FE 2A CC        111C 00 00 00 00
                        1120 00 00 00 00
```

Abb. 9.3-3 Der Maschinencode für den 8080 des gesamten Programms Labyrinth

man dabei am besten von dem in Abb. 9.2.2-6 dargestellten Sprung nach M 1 ausgeht und anstatt nach M 1 auf ein entsprechendes Hilfsprogramm springt.

Ein anderes Problem: Wenn die Maus auf ein Feld positioniert wird, an dessen Nachbarschaft keine Mauer grenzt, so irrt die Maus immer im Kreis herum. Man hat auch hier die Möglichkeit, etwas zu erfinden, das dies verhindert. Ferner kann auch ein Programm erfunden werden, das die Maus befähigt, einen einmal durchlaufenen Weg zu vereinfachen, und beim nächsten Mal den direkten Weg zu wählen.

9.4 Pawlowscher Hund

Wird einem Hund Fleisch vorgehalten, so beginnt die Produktion von Speichel und Magensaft. Um dem Tier einen bedingten Reflex beizubringen wird jedesmal, wenn dem Hund das Fleisch gegeben wird, zusätzlich eine Glocke geläutet. Wenn dies oft genug durchgeführt ist, braucht nur noch die Glocke geläutet zu werden, um die Produktion von Speichel und Magensaft zu bewirken. Wird dann allerdings zu oft allein die Glocke geläutet, ohne dem Hund das Fleisch zu geben, so erfolgt dieser Reflex nicht mehr. Das Tier hat gelernt, daß es „betrogen" wurde. Dieses Problem soll mit der Apparatur simuliert werden. *Abb. 9.4-1* zeigt ein Flußdiagramm, das dem Mikrocomputer die Eigenschaft gibt, dieses Verhalten darzustellen.

Zuerst soll die Bedienung beschrieben werden. „Fleisch" wird als Begriff eingetastet (F) und dann die Taste CR betätigt. Der Computer antwortet mit dem Ausdruck von „MS" was soviel bedeutet wie „Magensaft fließt". Wenn „Glocke" eingetastet (G) und CR betätigt wird, so passiert nichts weiter. Das bedeutet, der Zusammenhang zwischen Glocke und Fleisch wurde noch nicht gelernt. Dieser Lernvorgang kann simuliert werden, wenn „Fleisch" und „Glocke" in eine Zeile geschrieben werden und dann CR betätigt wird. Der Computer antwortet darauf mit MS, denn „Fleisch" wurde ja auch eingegeben. Diese Eingabe muß mehrfach wiederholt werden. Dann kann festgestellt werden, ob der Lernvorgang schon erfolgt ist. Das wird überprüft durch Eingabe von „Glocke" und CR. Wenn dann MS erscheint, so ist der Lernvorgang abgeschlossen. Wird jetzt aber mehrmals nur „Glocke" eingetastet ohne „Fleisch', so wird der Lernvorgang rückgängig. Es erscheint nicht mehr „MS" nach Eintasten von „Glokke". Der Lernvorgang kann aber sehr schnell erneut erfolgen, wenn „Glocke" und „Fleisch" wieder in eine Zeile geschrieben und CR betätigt wird.

Erklärung des Flußdiagramms:

Nach Start des Programms wird der hier mit „Lernvorgang" bezeichnete Zähler zunächst auf Null gesetzt. Anschließend werden die Register D und E auf Null gesetzt. Nun wird auf die Eingabe von einem Zeichen gewartet. Ist diese erfolgt, so findet eine Abfrage statt. Zuerst wird gefragt, ob das eingegebene Zeichen identisch mit dem Buchstaben F war. Falls es der Buchstabe F war, wird eine 1 in dem Register D abgespeichert.

Damit wird das Auftreten des Buchstabens F festgehalten und somit auch des Wortes „Fleisch".

Nach Abspeichern in D erfolgt der Sprung zurück und somit ein neues Warten auf ein nächstes Zeichen. Hat es sich nicht um den Buchstaben F gehandelt, so wird gefragt, ob es sich um den Buchstaben G handelte. Trifft dies zu, so wird das Erscheinen des Buchstabens G und damit des Wortes „Glocke" durch Abspeichern einer 1 in das Register E festgehalten. Anschließend erfolgt der Rücksprung und das Warten auf eine neue Eingabe. Hat es sich aber nicht um den Buchstaben G gehandelt, so erfolgt die Überprüfung auf den Befehl CR. Wenn dieser Befehl nicht identisch mit dem Zeichen war, so erfolgt ein Rücksprung und somit ein Warten auf eine neue Eingabe.

Wenn es der Befehl CR war, der eingegeben wurde, so muß nun überprüft werden, ob in der Zeichenfolge, die vor diesem CR eingegeben wurde, das Wort „Fleisch" oder mindestens der Buchstabe „F" vorgekommen war. Falls ja, so erfolgt der Ausdruck von MS. Anschließend wird in beiden Fällen überprüft, ob das Wort „Glocke" oder mindestens der Buchstabe „G" in der Zeichenfolge vorkam, die vor dem CR eingegeben wurde. Falls nein, so kann ein Sprung erfolgen, der diesmal an die Stelle erfolgt, wo zunächst die Register D und E gelöscht werden.

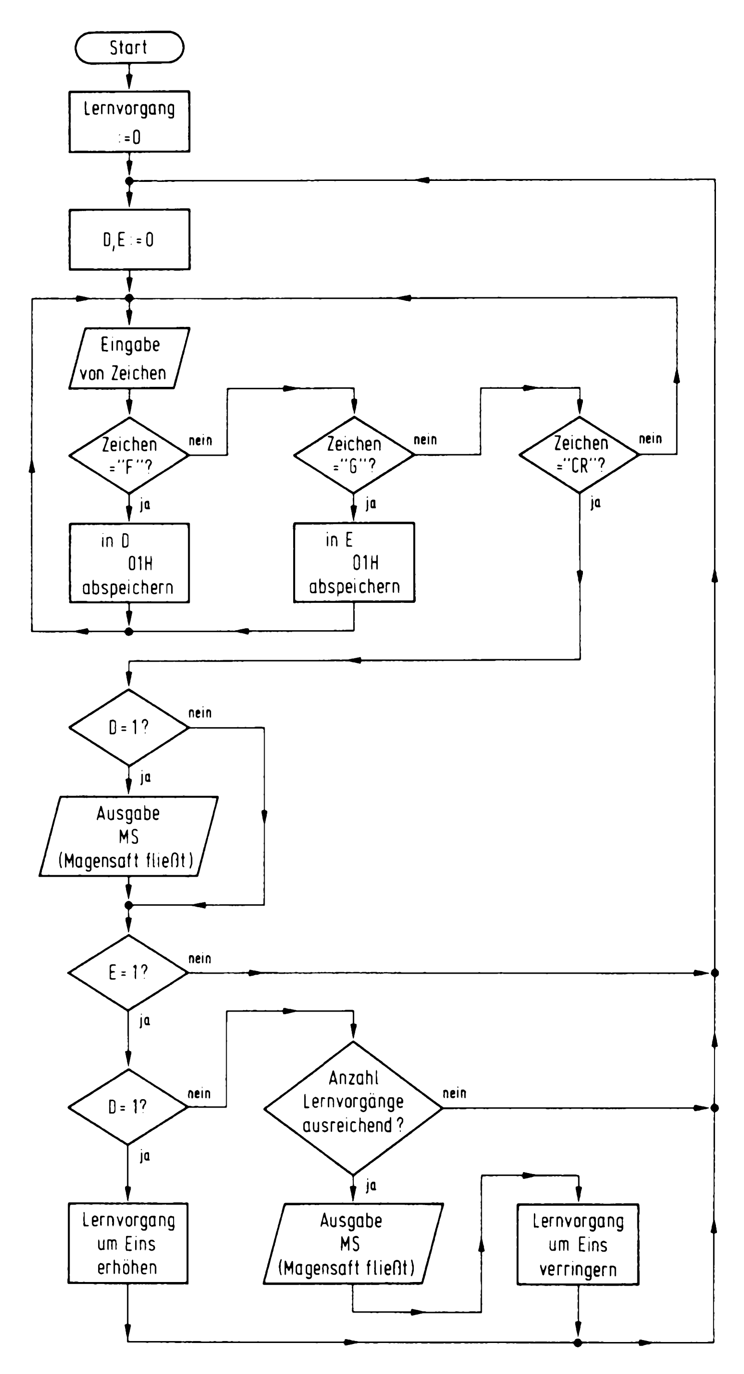


Abb. 9.4-1 Flußdiagramm des Programms „Pawlowscher Hund"

Trat „G“ aber auf, so wird das Zählregister „Lernvorgang“ um eins erhöht, wenn auch „F“ aufgetreten war. Danach erfolgt ein Rücksprung an die Stelle, wo zunächst die Register D und E zurückgesetzt werden. Ist das Zeichen „F“ aber nicht in der Zeichenfolge vor dem Befehl CR aufgetreten und das Zeichen „G“ ist aufgetreten, so erfolgt die Abfrage, ob schon der Zähler „Lernvorgang“ genügend hoch steht. Tut er dies nicht, erfolgt der Rücksprung zu der Stelle, an der die Register D und E auf Null gesetzt werden.

Ist aber der Zählerstand ausreichend hoch, d. h. der Lernvorgang beendet, so erfolgt der Ausdruck MS. Da nun aber kein „Fleisch“ gegeben wurde und nur die „Glocke“ geläutet wurde, erfolgt eine Erniedrigung des Lernvorgangzählers um eins. Danach erfolgt der Sprung zu der Stelle, an der die Register D und E auf Null gesetzt werden.

Detailprogramm Pawlowscher Hund mit Erweiterung:
Das Programm wird mit dem Befehl G 1319 CR gestartet.

```
1300 CD  CALL CI    ;Sub 1, eingeben
  01 FD             von Zeichen
  02 03

  03 4F  MOV C, A   ;vorbereiten für
                    Ausgabe
  04 CD  CALL       ;Zeichen ausgeben
  05 F4  ECHO
  06 01
  07 79  MOV A, C   ;In den
                    Akkumulator
                    zurückschaffen
  08 C9  RET        ;Zurück ins Haupt-
                    programm
1309 0E  MVI C, „M“
  0A 4D
  0B CD  CALL CO
  0C FA
  0D 03
  0E 0E  MVI C, „S“
  0F 53
  10 CD  CALL CO
  11 FA
  12 03
  13 0E  MVIC „CR“
  14 0D
  15 CD  CALL
  16 F4  ECHO
  17 01

  18 C9  RET
1319 21  LXI H,     ;Start des Programms
  1A 00  0000 H     rücksetzen
  1B 00


131C 11  LXI D,
  1D 00  0000 H
  1E 00

131F CD  CALL       ;Eingabe von
  20 00  SUB1       Zeichen
  21 13             Einsprung


  22 FE  CPI „F“    ;Vergleichen mit
  23 46             dem Zeichen F

  24 C2  JNZ 132CH  ;Falls nicht gleich
  25 2C             dem Zeichen
  26 13             ;weiter

  27 16  MVI D,     ;Falls gleich dem
  28 01  01H        Zeichen F, dann

                    ;1 auf D abspeichern
  29 C3  JMP 131FH  ;Anschließend Rück-
  2A 1F             sprung
  2B 13


132C FE  CPI „G“    ;Zeichen nun mit G
  2D 47             vergleichen


  2E C2  JNZ 1336H  ;Falls es das Zeichen
  2F 36             nicht ist, dann
  30 13             ;Abfrage fortsetzen
```

```
  31  1E  MVI E,     ;sonst 1 auf E ab-
  32  01  01H        speichern

  33  C3  JMP 131FH ;danach Rücksprung
  34  1F
  35  13
1336  FE  CPI „CR“   ;nun noch mit CR
  37  0D             vergleichen

  38  C2  JNZ 131FH ;Falls nicht erfüllt,
  39  1F            dann Rücksprung
  3A  13            ;und somit wieder-
                    holen

  3B  15  DCR D      ;Erkennen des Lern-
                     vorgangs und
  3C  C2  JNZ 1343H ;Weiterverarbeitung,
  3D  43            wenn Fleisch (F)
  3E  13            ;gegeben wurde,
                    dann nicht springen

  3F  CD  CALL       ;MS ausdrucken,
  40  09  SUB 2      da Fleisch(F)gegeben
  41  13             ;wurde

  42  14  INR D      ;Ausgangszustand
1343  1D  DCR E      ;Ertönte auch
                     die Glocke?

  44  C2  JNZ 131CH ;Wenn nicht,
  45  1C            dann Rücksprung
  46  13

  47  15  DCR D      ;Nur,wennFleisch(F)
                     und Glocke (G)

  48  C2  JNZ 134FH ;vorhandenwaren,er-
  49  4F            folgt der Lernvorgang
  A4  13            ;andernfalls Sprung
                    nach Überprüfung,
                    ;ob schon gelernt
                    wurde, da Glocke
                    da war
```

```
  4B  24  INR H      ;Lernvorgang durch-
                     führen

  4C  C3  JMP 131CH ;Danach Rücksprung
  4D  1C
  4E  13

134F  7C  MOV A, H  ;Ist schon ge-
                    nügend oft

  50  FE  CPI 08H    ;Glocke und Fleisch
  51  08             gleichzeitig
                     ;registriert worden
                     und somit der

  52  DA  JC 131CH   ;Lernvorgang abge-
  53  1C             schlossen?

  54  13             ;Falls nicht, Rück-
                     sprung

  55  CD  CALLSUB2;MS ausgeben
  56  09
  57  13
  58  25  DCR H      ;Da nur die Glocke
                     läutete, Schwächung
                     ;des Lernvorgangs
  59  C3  JMP 131CH
  5A  1C             ;Dann Rücksprung
  5B  13
```

9.5 Mathematisches Modell für „Leben–Sterben–Geboren werden“

Das Modell wurde von einem Mathematiker namens Conway entwickelt. Er wollte damit auf einfache Weise Lebensvorgänge simulieren. Als Lebensraum dient zum Beispiel der Bildschirm eines Datensichtgerätes. Dabei gelten folgende Regeln: Es überlebt derjenige, der auf acht Nachbarfeldern zwei oder drei Nachbarn hat.

Es stirbt, wer vier oder mehr Nachbarn hat, oder wer nur einen oder keinen Nachbarn hat.

Auf einem leeren Feld findet eine Geburt statt, wenn dieses Feld genau drei Nachbarn hat.
Conway wählte diese Regeln nach folgenden Forderungen:

1. Es soll kein Ausgangsmuster geben, für das es einen einfachen Beweis gibt, daß die Bevölkerung unbegrenzt wächst.

2. Es soll unbegrenzt wachsende Ausgangsmuster geben.

3. Es soll einfache Ausgangsmuster geben, die nach einer Wachstumsperiode wieder aussterben oder in festbleibende stabile oder oszillierende Phasen eintreten.
(Literatur: TV Computer 6800), [4].

Durch die Verwendung eines Bildschirms bzw. durch den begrenzten Speicherplatz tritt allerdings bei der praktischen Ausführung das Randfeld mehr oder weniger später in störende Erscheinung. Es soll dem Anwender überlassen bleiben, das Programm dafür zu entwickeln. Doch dazu ein paar Hinweise.

Man muß sich entscheiden, ob ein Programm erstellt wird, das mit dem Speicher des Datensichtgerätes arbeitet oder ob dazu der Speicher im 8080 verwendet wird. Die erste Methode bringt ein einfacheres Programm, das aber bei der Ausführung eine größere Zeit benötigt, die zweite Methode bringt ein aufwendigeres Programm, das aber viel schneller arbeitet.

(1) Bei der ersten Methode werden die acht Umfelder eines jeden Feldes abgefragt. Die Nachbarfelder, die besetzt sind, werden dann jeweils in ein Register eingelesen. Für jedes Feld kann dann eine Entscheidung gefällt werden,

a) ob das Lebewesen stirbt,

b) ob das Wesen leben bleibt, oder

c) ob ein neues Lebewesen geboren wird, falls das Feld vorher nicht besetzt war und die Bedingungen bezüglich der 3 Nachbarn erfüllt waren, die noch notwendig sind.

Nun kann aber ein Lebewesen, das stirbt, nicht einfach im Speicher gelöscht werden, denn sonst könnte sich eine Verfälschung des Ablaufs im Prozeß ergeben. Es gibt aber eine einfache Möglichkeit, diese Schwierigkeit zu umgehen. Die Lebenden werden z. B. mit dem Buchstaben L bezeichnet. Die Felder, auf denen die Lebenden sterben werden, werden zunächst mit S benannt. Und Lebewesen, die geboren werden, werden mit dem Zeichen G beschrieben. Beim Durchlauf für die Berechnung werden für die Zählung der Nachbarn sowohl mit L als auch mit S besetzte Felder gezählt. Wurden die Berechnungen für den gesamten Lebensraum gemacht und nach diesen Regeln entsprechende Umbesetzungen, so erfolgt ein zweiter Durchlauf, wobei alle Felder mit L unverändert gelassen werden, alle Felder mit S gelöscht werden. Sie werden mit dem Leerzeichen überschrieben. Alle Felder mit G werden in Felder mit L umbesetzt. Danach ist ein vollständiger Durchlauf beendet. Ein neuer Durchlauf kann nun angeschlossen werden.

(2) Bei der zweiten Methode ist es nötig, im Hauptspeicher mehrere Felder festzulegen. Die Felder müssen allerdings nicht mit 1 KByte belegt werden. Da für die Entscheidung „belegt" – „nicht belegt" ein bit genügt, ist es nötig, nur ein Feld von je 128 Bytes festzulegen. Mehrere Felder werden benötigt, um bei der Berechnung die „Sterbenden" und die „Geborenen" festzuhalten. Es sind somit genau drei Felder mit je 128 Bytes nötig.

Dabei ist vorausgesetzt, daß der ganze Bildschirm des Datensichtgerätes in der Ausbaustufe von 1 KByte als Lebensraumfläche verwendet wird. Das Programm nach dieser 2. Methode benötigt einen Eingabeprogrammteil, ein Ausgabeprogrammteil und das eigentliche Programm. Wie schon gesagt, ist dieser zweite Weg erheblich umfangreicher. Doch wird der Aufwand mit einem viel schnelleren Programmablauf belohnt.

9.6 Primzahlberechnung

Es soll wieder ein ausgeführtes Programm besprochen werden. Ziel dieses Programms ist es, Primzahlen zu berechnen. Dabei werden der Einfachheit halber folgende Einschränkungen gemacht. Die Zahlen können bis maximal 255 dargestellt werden. Die Darstellung der Primzahlen erfolgt im Dualsystem.

Dadurch wurde vermieden, eine umfangreiche Arithmetik zu entwickeln.

Das Verfahren arbeitet etwa wie folgt:

Zunächst wird bei einem Anfangswert begonnen, der eine ungerade Zahl darstellt. Dieser Wert wird auf die Eigenschaft einer Primzahl untersucht, in dem dieser Wert durch alle Zahlen geteilt wird, die kleiner sind als die Hälfte dieses Wertes. Ergibt sich bei allen Divisionen kein Ergebnis, d. h. die Teilung geht nicht auf, so handelt es sich um eine Primzahl. Geht eine der Divisionen auf, so war es keine Primzahl und die nächste ungerade Zahl wird untersucht. Im Falle einer Primzahl wird diese ausgedruckt. Die nicht als Primzahl gefundenen Zahlen werden nicht ausgedruckt.

Abb. 9.6-1 zeigt den Hauptteil dieses Programms.

Es wird zunächst nach Start des Programms der Inhalt des Registers B mit dem Wert 1 belegt.

Dann erfolgt das Kopieren dieses Wertes in das Register H. Das Register B stellt dabei einen Zähler dar. Er beinhaltet die zu untersuchende Zahl. Es erfolgt dann noch ein Kopieren des Inhalts des Registers B in das Register A. Dann folgt der Befehl RRC. Der Inhalt des Akkumulators wird bit für bit um eine Position nach rechts geschoben. Dies entspricht im dualen Zahlensystem einer arithmetischen Division durch zwei. Dann wird mit dem Befehl ANI 7FH eine UND-Verknüpfung des im Register A vorhandenen Wertes mit dem Wert 3FH durchgeführt. Dadurch wird die erste Stelle des Wortes in Register A auf jeden Fall 0, denn durch den Befehl RRC gelangt die letzte, geringwertigste Stelle des Wortes in das erste Byte. Der Wert des Registers A wird dann in das Register D kopiert. Die Division wird hier durch eine fortlaufende Subtraktion ersetzt. Dabei ergibt sich eine weitere Vereinfachung.

Der Zähler enthält beispielsweise den Wert 13. Nun wird zur Untersuchung auf die Eigenschaft, ob es sich um eine Primzahl handelt, dieser Wert des Zählers durch 2 geteilt. Es ergibt sich ein Wert 6.5. Es wäre nun notwendig, festzustellen, ob ein Rest besteht. Dazu müßte erst einmal der Dezimalteil abgeschnitten werden und dafür wäre ein spezielles Unterprogramm nötig. Dann könnte erst festgestellt werden, ob ein Rest vorhanden (≠0) oder kein Rest (=0) vorhanden ist. Bei dem hier benutzten Verfahren geschieht die Überprüfung jedoch wie folgt:

Der Divisor wird zunächst von der zu untersuchenden Zahl abgezogen. Dann erfolgt eine Überprüfung des neuen Wertes. Ist dieser größer als Null, so erfolgt ein Sprung, und der Divisor wird erneut abgezogen. Ist der Wert gleich Null, so ergab die Division keinen Rest. Ist der Wert kleiner als Null, so ist ein Rest vorhanden. Es wird als erstes der Wert des Zählers aus dem Register H in das Register A kopiert. Dann wird der Registerinhalt von A um den Inhalt des Registers E vermindert. Und anschließend wird das Ergebnis in das Register H zurückgespeichert. Nun erfolgt die Abfrage. Ist der Wert des Registers gleich Null, so erfolgt ein Sprung, der eine zweimalige Erhöhung des Registerinhaltes von B bewirkt und somit einen neuen „Prim"-Zähler erzeugt. Dann erfolgt der Sprung zur Marke A und das Programm läuft von dort aus neu weiter.

Andernfalls erfolgt eine Abfrage, ob der Inhalt des Registers A kleiner als Null ist. Ist dies nicht der Fall, so muß der Inhalt des Registers A größer als Null sein, d. h. nicht kleiner und nicht gleich Null. Es erfolgt ein Sprung zu der Marke B. Andernfalls wird der Subtrahend (bzw. Divisor) in das Register A von dem

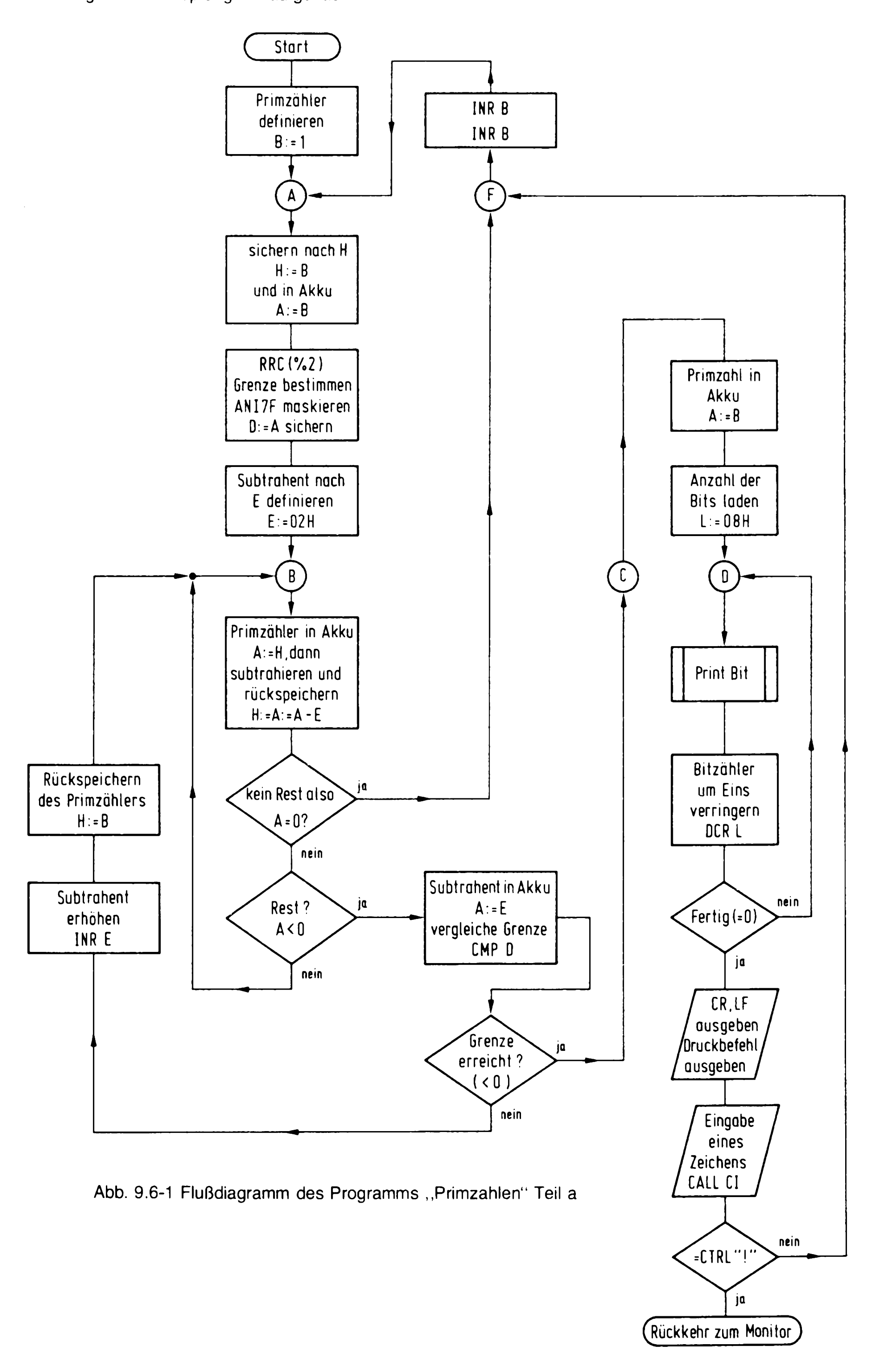


Abb. 9.6-1 Flußdiagramm des Programms „Primzahlen" Teil a

Register E kopiert, und es wird ein Vergleich mit dem Register D durchgeführt. Sind die beiden Werte identisch, so erfolgt ein Sprung zur Marke C, und es handelt sich um eine Primzahl. Andernfalls, wenn also die Grenze, die durch den Inhalt des Registers D bestimmt ist, noch nicht erreicht wurde, so wird der Inhalt des Registers E erhöht, das den Subtrahend enthält. Dann erfolgt eine Rückspeicherung des Wertes des „Prim"-Zählers in das Register H.

Nun zu dem mit Marke C bezeichneten Programmteil.

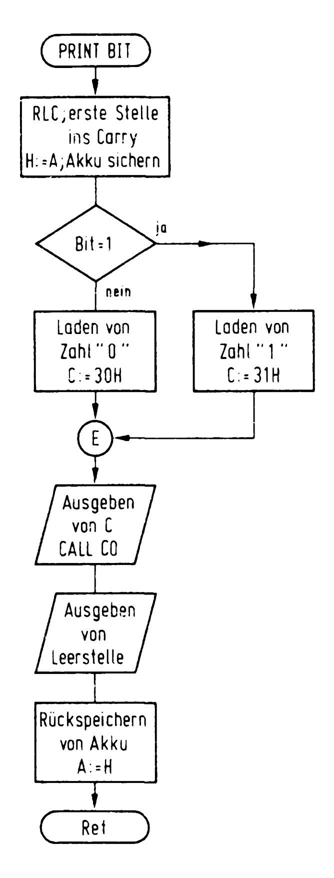


Abb. 9.6-2 Flußdiagramm des Programms „Primzahlen" Teil b

Dieses Programmteil hat die Aufgabe, die Primzahl auf dem Datensichtgerät in dualer Form auszugeben und dann auch noch einen Druckbefehl zu geben, so daß ein angeschlossener Drucker die Primzahl ausdrucken kann. Dabei wurde das Format des Ausdrucks für einen Drucker mit 16 Zeichen vorgesehen. *Abb. 9.6-2* zeigt das dazugehörige Unterprogramm PRINT BIT.

Abb. 9.6-3 zeigt den Ausdruck, wie es das Programm liefert. Das Programm befördert dazu zunächst die Primzahl in das Register A mit dem Befehl A:=B.

Dann wird die feststehende Anzahl von Bits mit dem Befehl L:=08H in den Speicher geladen. Dann folgt der Aufruf eines noch zu besprechenden Unterprogramms, das die höherwertige Stelle des Inhalts des Registers A bitweise nach links verschiebt. Der Bitzähler in Register L wird um eins verringert.

Ist dieser noch nicht Null, so erfolgt ein Rücksprung zur Marke D, andernfalls wird CR, LF ausgegeben und der Druckbefehl (03H) wird ebenfalls gegeben. Danach wird die Eingabe eines Zeichens vom Benutzer verlangt. Dadurch kann erstens die Ausdruckgeschwindigkeit bestimmt werden. Zweitens ist es möglich, den Druckvorgang zu beenden, indem die Tasten „CTRL!" betätigt werden.

Es erfolgt dann nämlich ein Rücksprung in das Monitorprogramm. Andernfalls, also wenn ein beliebiges anderes Zeichen eingegeben wurde, erfolgt ein Sprung zur Marke F und somit eine Fortsetzung der Berechnung der Primzahlen.

Nun noch die Besprechung des Unterprogramms PRINT BIT.
Als erster Befehl folgt RLC. Damit wird der Inhalt des Registers A nach links verschoben und die erste Stelle des Wortes, also die höherwertige Stelle, gelangt sowohl an die letzte Stelle des Wortes als auch in den Übertragsspeicherplatz. Der Inhalt des Registers A wird dann in den Speicher H kopiert. Nun befindet

```
1 1 1 1 1 0 1 1        0 1 1 0 0 1 1 1
1 1 1 1 0 0 0 1        0 1 1 0 0 1 0 1
1 1 1 0 1 1 1 1        0 1 1 0 0 0 0 1
1 1 1 0 1 0 0 1        0 1 0 1 1 0 0 1
1 1 1 0 0 1 0 1        0 1 0 1 0 0 1 1
1 1 1 0 0 0 1 1        0 1 0 0 1 1 1 1
1 1 0 1 1 1 1 1        0 1 0 0 1 0 0 1
1 1 0 1 0 0 1 1        0 1 0 0 0 1 1 1
1 1 0 0 0 1 1 1        0 1 0 0 0 0 1 1
1 1 0 0 0 1 0 1        0 0 1 1 1 1 0 1
1 1 0 0 0 0 0 1        0 0 1 1 1 0 1 1
1 0 1 1 1 1 1 1        0 0 1 1 0 1 0 1
1 0 1 1 0 1 0 1        0 0 1 0 1 1 1 1
1 0 1 1 0 0 1 1        0 0 1 0 1 0 1 1
1 0 1 0 1 1 0 1        0 0 1 0 1 0 0 1
1 0 1 0 0 1 1 1        0 0 1 0 0 1 0 1
1 0 1 0 0 0 1 1        0 0 0 1 1 1 1 1
1 0 0 1 1 1 0 1        0 0 0 1 1 1 0 1
1 0 0 1 0 1 1 1        0 0 0 1 0 1 1 1
1 0 0 1 0 1 0 1        0 0 0 1 0 0 1 1
1 0 0 0 1 0 1 1        0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 1        0 0 0 0 1 1 0 1
1 0 0 0 0 0 1 1        0 0 0 0 1 0 1 1
0 1 1 1 1 1 1 1        0 0 0 0 0 1 1 1
0 1 1 1 0 0 0 1        0 0 0 0 0 1 0 1
0 1 1 0 1 1 0 1        0 0 0 0 0 0 1 1
0 1 1 0 1 0 1 1        0 0 0 0 0 0 0 1
```

Abb. 9.6-3 Ausdruck, wie er von dem Primzahlprogramm geliefert wird

sich die Information über die zu druckende Stelle im Übertragsbit. Es erfolgt eine direkte Abfrage des Bits mit einer entsprechenden Sprunganweisung. Ist das Bit gesetzt, so wird in das Register C der Wert 31H übertragen, andernfalls der Wert 30H in das Register C. Der Wert 31H bedeutet in dem verwendeten ISO-7-bit-Code das Zeichen 1 und der Wert 30H das Zeichen 0. Dieser Wert wird dann mit dem üblichen Befehl CALL CO ausgegeben. Dann erfolgt noch das Ausgeben einer Leerstelle, z. B. durch die Befehlsfolge MVI C, 20H ;CALL CO. Schließlich wird der Registerinhalt H in das Register A kopiert, um so eine Weiterverarbeitung zu ermöglichen. Es erfolgt dann der obligatorische Rücksprungbefehl RET.

Abb. 9.6-4 zeigt den Maschinencode für dieses Programm. Dabei wird das Programm bei der Adresse 1200H des Hauptspeichers des SDK 80 KIT geladen. Der Start des Programms erfolgt dann mit der Befehlsfolge G 1200 CR.

Druckprogramm

Da oft auch Interesse für das Programm besteht, das den Ausdruck eines bestimmten Programms ermöglicht, soll das erste solche Dienstprogramm beschrieben werden.

Abb. 9.6-5 zeigt das dazugehörige Flußdiagramm.

Nach Start des Programms wird das Registerpaar als erstes mit der Endadresse des auszudruckenden Bereichs besetzt.

```
1200 06 11FF CF
1201 01 1200 06
1202 60 1201 01
1203 78 1202 60
1204 0F 1203 78
1205 E6 1204 0F
1206 7F 1205 E6
1207 57 1206 7F
1208 1E 1207 57
1209 02 1208 1E
120A 7C 1209 02
120B 93 120A 7C
120C 67 120B 93
120D CA 120C 67
120E 3E 120D CA
120F 12 120E 3E
1210 D2 120F 12
1211 0A 1210 D2
1212 12 1211 0A
1213 7B 1212 12
1214 BA 1213 7B
1215 D2 1214 BA
1216 1D 1215 D2
1217 12 1216 1D
1218 1C 1217 12
1219 60 1218 1C
121A C3 1219 60
121B 0A 121A C3
121C 12 121B 0A
121D 78 121C 12
121E 2E 121D 78
121F 08 121E 2E
1220 CD 121F 08
1221 43 1220 CD
1222 12 1221 43
1223 2D 1222 12
1224 C2 1223 2D
1225 20 1224 C2
1226 12 1225 20
1227 0E 1226 12
1228 0D 1227 0E
1229 CD 1228 0D
122A FA 1229 CD
122B 03 122A FA
122C 0E 122B 03
122D 0A 122C 0E
122E CD 122D 0A
122F FA 122E CD
1230 03 122F FA
1231 0E 1230 03
1232 03 1231 0E
1233 CD 1232 03
1234 FA 1233 CD
1235 03 1234 FA
1236 CD 1235 03
1237 FD 1236 CD
1238 03 1237 FD
1239 FE 1238 03
123A 01 1239 FE
123B CC 123A 01
123C 08 123B CC
123D 00 123C 08
123E 04 123D 00
123F 04 123E 04
1240 C3 123F 04
1241 02 1240 C3
1242 12 1241 02
1243 07 1242 12
1244 67 1243 07
1245 DA 1244 67
1246 4D 1245 DA
1247 12 1246 4D
1248 0E 1247 12
1249 30 1248 0E
124A C3 1249 30
124B 4F 124A C3
124C 12 124B 4F
124D 0E 124C 12
124E 31 124D 0E
124F CD 124E 31
1250 FA 124F CD
1251 03 1250 FA
1252 0E 1251 03
1253 20 1252 0E
1254 CD 1253 20
1255 FA 1254 CD
1256 03 1255 FA
1257 7C 1256 03
1258 C9 1257 7C
2-.G13001258 C9
```

Abb. 9.6-4 Maschinencode des Primzahlprogramms

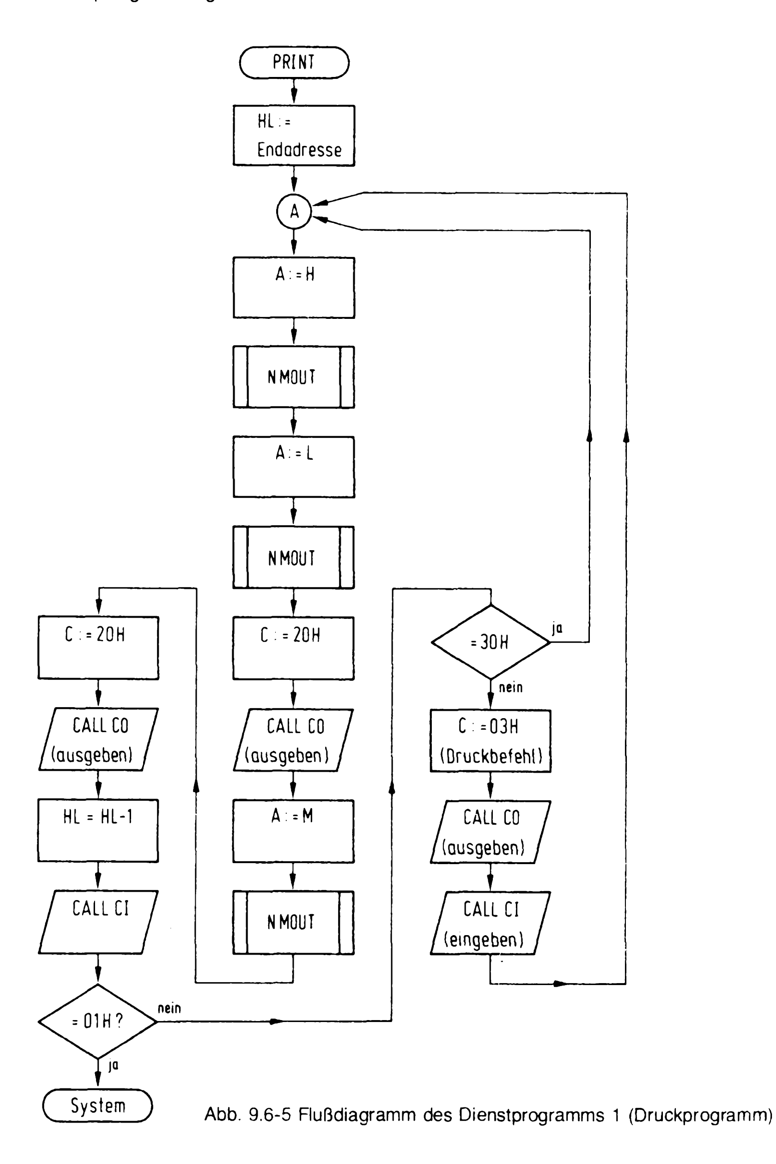


Abb. 9.6-5 Flußdiagramm des Dienstprogramms 1 (Druckprogramm)

Die Frage ist berechtigt: „Warum nicht die Anfangsadresse?“ Das hat folgenden Grund. Der verwendete, in Kapitel 5 schon erwähnte Drucker hat einen Papiervorschub von oben nach unten.Damit sich ein gut lesbares Bild ergibt, ist es notwendig, den Ausdruck des Speicherinhalts auch umgekehrt vorzunehmen, so daß bei der Endadresse des darzustellenden Bereichs begonnen wird. Wenn ein Drucker vorhanden ist, der die Information von oben nach unten auf das Papier bringt, also einen Papiervorschub nach oben besitzt, so kann das Programm durch Änderung eines einzigen Befehls, wie noch besprochen wird, veranlaßt werden, den Speicherinhalt entsprechend auszulesen. Ist das Registerpaar HL mit der End-

adresse (oder nach Änderung des einen Programmbefehls mit der Anfangsadresse) belegt, so wird der Inhalt des Registers H in das Register A kopiert.

Dann wird ein Unterprogramm mit dem Namen NMOUT aufgerufen, das sich im Monitorpogramm befindet. Es hat die Eigenschaft, den im Register A enthaltenen Wert in zwei Zeichen des ISO-7-bit-Code umzuwandeln und auszugeben, die den Wert des Registers A in hexadezimaler Schreibweise wiedergeben. Anschließend wird der Inhalt des Registers L in das Register A kopiert. Dann wird dieser Inhalt entsprechend mit dem Aufruf des Unterprogramms NMOUT in hexadezimaler Schreibweise ausgegeben. Nun wurde erreicht, daß an den Anfang der auszudrukkenden Zeile die Adresse des auszudruckenden Speichers geschrieben wird. Anschließend wird das Register C mit dem Wert 20H besetzt und es erfolgt eine Ausgabe dieses Wertes mit dem Unterprogramm CO. Damit wurde ein Leerzeichen ausgegeben.

Es wird nun der Inhalt der Speicherzelle in das Register A übertragen, dessen Adresse durch den Inhalt des Registerpaars HL bestimmt ist. Dann wird mit Hilfe des Unterprogramms NMOUT der Wert des Registers A in hexadezimaler Schreibweise ausgegeben. Anschließend wird dem Register C der Wert 20H zugewiesen, und mit dem Befehl CALL CO wird dieser Wert, also ein Leerzeichen, ausgegeben. Es folgt die Erniedrigung des Wertes des Registerpaars HL um eins. Damit wird die nächste Adresse bestimmt. Ist ein Drucker vorhanden, der die Zeilen „normal" ausdruckt, so kann er durch das Ersetzen des Befehls HL:=HL-1 (DCX H) durch den Befehl HL:=HL+1 (INX H) einen entsprechenden Ausdruck erhalten.

Jetzt erfolgt der Befehl CALL CI. Damit wird auf eine Eingabe vom Benutzer gewartet. Gibt dieser den Wert 01H durch Betätigen der Tasten „CTRL !" ein, so erfolgt ein Rücksprung in das System, also in das Monitorprogramm. Andernfalls wird das eingegebene Zeichen noch mit dem Zeichen „0" (30H) verglichen. Stimmen die Zeichen überein, dann erfolgt ein Sprung zur Marke A und die nächste Adresse wird mit dem Inhalt des dazugehörigen Speicherplatzes ausgedruckt, ohne zuvor einen Druckbefehl auszugeben. Wird ein anderes Zeichen eingegeben, so erfolgt die Ausgabe des Druckbefehls durch die Zuweisung des Wertes 03H an das Register C und ein anschließendes Ausgeben des Wertes mit dem Befehl CALL CO. Danach folgt wieder ein Eingabeaufruf, womit erreicht werden soll, daß bei einem langsamen Druckwerk keine Störungen auftreten. Durch Betätigen einer beliebigen Taste nach dem Druckvorgang erfolgt die Ausgabe der nächsten Speicherzelle durch Sprung auf Marke A, wie schon beschrieben. Den Maschinencode für dieses Programm zeigt *Abb. 9.6-6*, wobei das Programm durch die Befehlsfolge G 1300 CR gestartet werden kann. Auf den Speicherzellen 1301H und 1302H steht die Endadresse (bzw. Anfangsadresse). Diese muß vor dem Start entsprechend dem gewünschten Speicherabschnitt, der dargestellt werden soll, eingegeben werden.

Beispiel: Ein Programm befindet sich auf den Speicherplätzen 1200H...1234H. Es soll ausgedruckt werden.

Zunächst wird mit dem Befehl I 1301 CR 34 12 ESC die Adresse (Endadresse) eingegeben. Dann wird das Programm durch einen Befehl G 1300 CR gestartet.

9.7 Aufstellen von Funktionstabellen

Hier soll ein Programm erklärt werden, das für die Vereinfachung von digitalen Schaltungen nützlich sein kann. Das Programm gestattet die Umwandlung einer gegebenen Funktionstabelle einer logischen Verknüpfung mit maximal 4 Eingängen in die Darstellung durch ein Karnaughdiagramm. Es ist möglich, manuell

```
1300 21 12FF EF
1301 32 1300 21
1302 13 1301 32
1303 7C 1302 13
1304 CD 1303 7C
1305 C3 1304 CD
1306 02 1305 C3
1307 7D 1306 02
1308 CD 1307 7D
1309 C3 1308 CD
130A 02 1309 C3
130B 0E 130A 02
130C 20 130B 0E
130D CD 130C 20
130E FA 130D CD
130F 03 130E FA
1310 7E 130F 03
1311 CD 1310 7E
1312 C3 1311 CD
1313 02 1312 C3
1314 0E 1313 02
1315 20 1314 0E
1316 CD 1315 20
1317 FA 1316 CD
1318 03 1317 FA
1319 2B 1318 03
131A CD 1319 2B
131B FD 131A CD
131C 03 131B FD
131D FE 131C 03
131E 01 131D FE
131F CC 131E 01
1320 08 131F CC
1321 00 1320 08
1322 FE 1321 00
1323 30 1322 FE
1324 CA 1323 30
1325 03 1324 CA
1326 13 1325 03
1327 0E 1326 13
1328 03 1327 0E
1329 CD 1328 03
132A FA 1329 CD
132B 03 132A FA
132C CD 132B 03
132D FD 132C CD
132E 03 132D FD
132F C3 132E 03
1330 03 132F C3
1331 13 1330 03
1332 00 1331 13
 *.G13001332 00
```

Abb. 9.6-6 Maschinencode des Dienstprogramms 1

mit den Regeln entsprechend dem Karaughdiagramm Vereinfachungen der eingegebenen logischen Verknüpfungen vorzunehmen, falls nicht schon die einfachste Version vorhanden ist. Die Eingabe der Funktionstabelle geschieht dabei folgendermaßen. Der Computer gibt den Code für die vier Eingänge an und der Benutzer muß das Ausgangssignal eingeben. Dann gibt der Computer die nächste Eingangskombination an, und der Benutzer tastet wieder das dazugehörige Ausgangssignal ein. Dies geschieht solange, bis alle 16 Möglichkeiten eingegeben wurden.

Die Umwandlung in ein Karnaughdiagramm erfolgt wie es *Abb. 9.7-1* zeigt. In das Karnaughdiagramm sind die Adressen der Zeilen der Funktionstabelle eingetragen.

Programmbeschreibung

Abb. 9.7-2 zeigt das erste Unterprogramm, das besprochen werden soll. Es hat vor allem die Aufgabe, einen Wert, der im Register C steht, auszugeben. Dies geschieht mit dem nun schon bekannten Befehl CALL CO. Außerdem muß der Inhalt des Registers A gewahrt bleiben. Also ist es notwendig, den Inhalt des Registers A vor Ausführung des Befehls CALL CO zu erhalten. Dies geschieht mit dem Befehl PUSH PSW, der den Wert des Registers A zusammen mit der Statusinformation auf den Stapelspeicher schafft. Die Ausführung des Befehls CALL CO nämlich würde eine Veränderung des Registerinhalts A mit sich bringen. Anschließend nach der Ausführung des Befehls CALL CO wird der ursprüngliche Inhalt

Nr	E_1	E_2	E_3	E_4	/	A
0	0	0	0	0	/	×
1	0	0	0	1	/	×
2	0	0	1	0	/	×
3	0	0	1	1	/	×
4	0	1	0	0	/	×
5	0	1	0	1	/	×
6	0	1	1	0	/	×
7	0	1	1	1	/	×
8	1	0	0	0	/	×
9	1	0	0	1	/	×
A	1	0	1	0	/	×
B	1	0	1	1	/	×
C	1	1	0	0	/	×
D	1	1	0	1	/	×
E	1	1	1	0	/	×
F	1	1	1	1	/	×

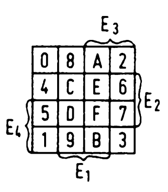


Abb. 9.7-1 Funktionstabelle und Anordnung der Werte in Karnaughdiagrammdarstellung

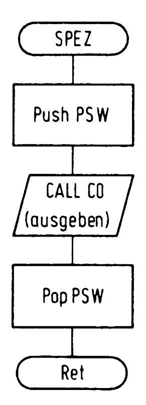


Abb. 9.7-2 Unterprogramm SPEZ

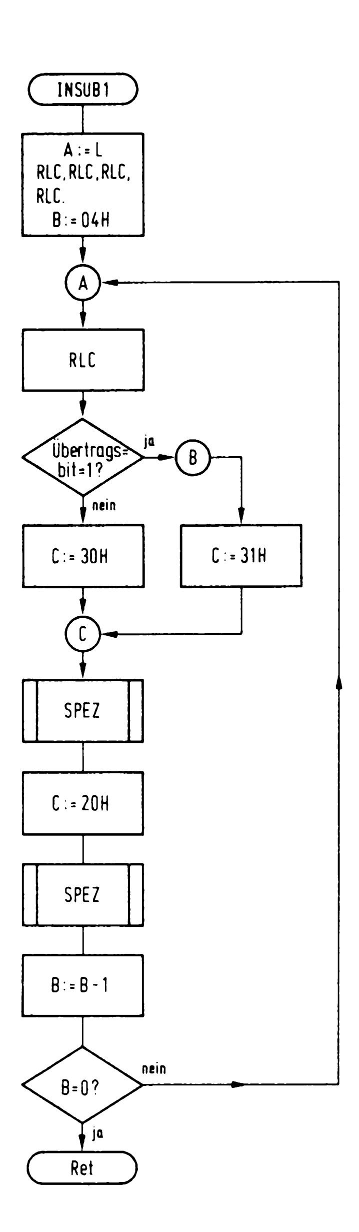


Abb. 9.7-3 Unterprogramm INSUB 1

des Registers A mit dem Befehl POP PSW wieder vom Stapelspeicher zurückgeschafft.

Dann folgt der Rücksprungbefehl.

Dieses Unterprogramm wird wiederum für ein anderes Unterprogramm verwendet. Es ist das Unterprogramm INSUB 1, das *Abb. 9.7-3* zeigt. Dieses Programm hat die Aufgabe, die ersten vier bit eines im Register L stehenden Wertes bit für bit auszugeben. Das Register L beinhaltet nämlich den Code für die vier Eingänge der einzugebenden logischen Verknüpfung. Der Code soll dual ausgegeben werden. Nach Start des Unterprogramms INSUB 1 wird der Inhalt des Registers L als erstes in das Register A kopiert. Dort wird der Wert bitweise viermal nach links verschoben. Anschließend wird nach dem Register B der Wert 04H zugewiesen.

Bei der Marke A erfolgt die erneute Verschiebung nach links. Damit wird die erste Bit-Stelle des auszugebenden Wertes in das Übertragsbit geschafft, natürlich auch in die geringwertigste Stelle des Bytes, was aber nicht weiter ausgenutzt wird. Es erfolgt die Abfrage, ob das Übertragsbit gesetzt ist oder nicht. Ist es gesetzt, so erfolgt die Zuweisung des Wertes 31H, der das Zeichen „1“ im ISO-7-bit-Code darstellt, an das Register C.

Ist das Übertragsbit nicht gesetzt, so erfolgt die Zuweisung des Wertes 30H also das Zeichen „0“ an das Register C.

Es wird bei Marke C fortgefahren. Es erfolgt der Aufruf des Unterprogramms SPEZ, das ja die Aufgabe hat, den Wert des Registers C entsprechend auszugeben. Der Wert des Registers A bleibt erhalten. Dem Register C wird jetzt der Wert 20H, der das Zeichen „space“ darstellt, zugewiesen, sowie anschließend durch Aufruf des Unterprogramms SPEZ ausgegeben. Es erfolgt die Erniedrigung des Inhalts des Registers B um den Wert eins. Dann folgt die Abfrage, ob das Register B den Wert 0 erhielt. Wenn nein, so erfolgt ein Sprung zur Marke A und der Programmabschnitt wird erneut durchgearbeitet. Andernfalls wurde der Programmteil insgesamt viermal durchlaufen und es erfolgt ein Rücksprung.

Das Unterprogramm INA hat, wie die *Abb. 9.7-4* zeigt, die Aufgabe auf eine Eingabe vom Benutzer zu warten und das eingegebene Zeichen als Kontrolle wieder auszugeben. Dazu erfolgt zunächst die Ausführung des Befehls CALL CI, dann erfolgt die Zuweisung des Inhalts des Registers A an das Register C, und schließlich wird mit dem Befehl CALL CO das Zeichen wieder ausgegeben. Dann erfolgt der Rücksprung.

Abb. 9.7-5 zeigt das Unterprogramm IN MAIN. Es ruft insgesamt alle vorher beschriebenen Unterprogramme auf. Es hat die Aufgabe, die Belegungen der Ausgangswerte der Funktionstabelle vom Benutzer anzufordern und diese abzuspeichern.

Die Werte werden in den Hauptspeicher geschafft, in dem Beispiel auf die Speicherstellen beginnend mit der Adresse 1100H. Dem Registerpaar HL wird deshalb zunächst der Wert 1100H zugewiesen. Das Register D wurde als Zähler für die Anzahl der Werte verwendet. Ihm wird der Wert 10H zugewiesen. 10H ist der hexadezimale Wert, der dezimal genau (10H =) 16 entspricht. Es folgt bei der Mar-

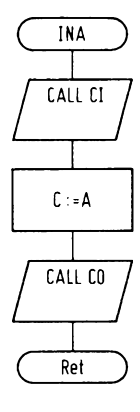


Abb. 9.7-4 Unterprogramm IN A

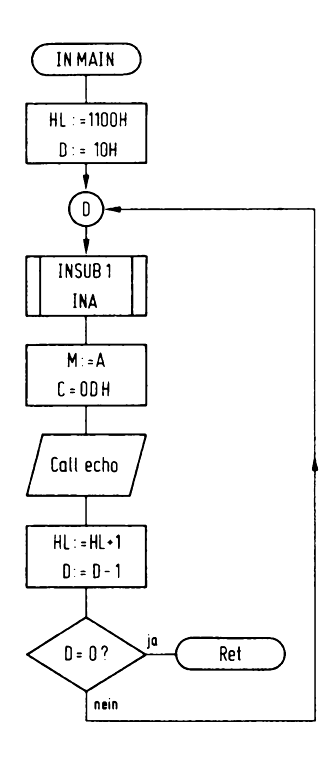


Abb. 9.7-5 Unterprogramm IN MAIN

folge CR, LF ausgegeben. Dies bewirkt bekanntlich, daß der Cursor in die nächste Zeile auf die linke Seite positioniert wird. Es folgt als nächstes die Erhöhung des Inhalts des Registerpaars HL um eins. Damit wird die nächste Speicherzelle bestimmt. Der Inhalt des Registers D wird um eins erniedrigt. Ist der Inhalt des Registers D Null geworden, so erfolgt ein Rücksprung, andernfalls wurden noch nicht alle Werte eingegeben, und es erfolgt ein Sprung zur Stelle D.

Das Unterprogramm WAIT, das *Abb. 9.7-6* zeigt, hat die Aufgabe, eine Verzögerung zu bewirken. Wird beim Datensichtgerät der Befehl „clear" gegeben, so kann es sein, daß die Ausführung dieses Befehls länger dauert, als der Zeit zwischen der Ausgabe des Befehls „clear" und der Ausgabe eines nächsten Zeichens „clear" an das Datensichtgerät entspricht. Dadurch würden Fehlinformationen entstehen. Nach der Ausgabe des Befehls „clear" wird also das Unterprogramm WAIT eingeschaltet und somit eine Störung vermieden.

ke D der Aufruf des Unterprogramms INSUB 1. Damit wird der Inhalt des Registers L dual ausgegeben (die 4 geringwertigen Dualstellen). Anschließend wird durch Aufruf des Unterprogramms INA der Ausgangswert dieser Eingangskombination eingegeben. Im ersten Fall des Durchlaufs der Schleife ist der zuvor ausgegebene Wert des Registers L gleich 0000. ES wird mit dem Befehl M: = A der Inhalt des Registers A in die Speicherzelle geschafft, deren Adresse durch den Inhalt des Registerpaars HL bestimmt ist.

Darauf folgt die Zuweisung des Wertes 0DH an das Register C. Anschließend folgt der Aufruf eines Unterprogramms mit dem Namen ECHO. Dieses Programm befindet sich ebenfalls im Monitor. Wenn im Register C das Zeichen CR (Code 0DH) steht, so wird nach Aufruf des Unterprogramms ECHO die Zeichen-

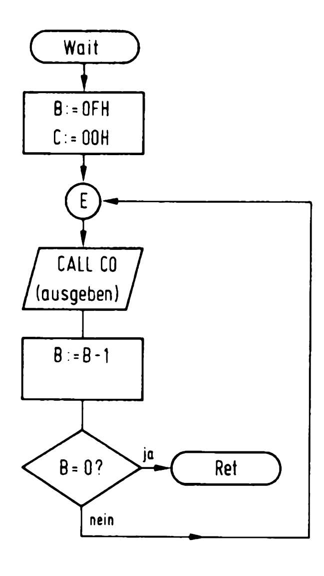


Abb. 9.7-6 Unterprogramm WAIT SUB

Nach Aufruf dieses Unterprogramms wird dem Register B zuerst der Wert 0FH zugewiesen. Dieser Wert bestimmt die Verzögerungszeit. Dem Register C wird der Wert 00 zugewiesen, dessen Bedeutung im ISO-7-bit-Code eine Nulloperation darstellt. Es folgt nun die Ausgabe des Inhalts des Registers C an das Datensichtgerät. Damit wird eine weitere Verzögerung erreicht. Die Ausgabe bleibt aber für das Datensichtgerät selbst ohne Bedeutung. Nun wird der Inhalt des Registers B um eins verringert. Es folgt anschließend ein Sprung zur Marke E, falls der Inhalt des Registers B noch nicht Null geworden ist, andernfalls erfolgt der Rücksprung.

Ausgabeteil
Der Ausgabeteil hat die Aufgabe, das Karnaughdiagramm auszugeben. Dazu müssen die Ausgangswerte der Funktionstabelle in anderer Reihenfolge ausgegeben werden. Die Umverteilung geschieht mit Hilfe einer Liste, die bei der hier gewählten Adresse auf dem Speicherplatz mit der Adresse 1150H beginnt.

Das Unterprogramm OUT1, dessen Flußdiagramm *Abb. 9.7-7* zeigt, hat die Aufgabe, den Ausgangswert der Funktionstabelle auszugeben, der durch den Inhalt der Liste bestimmt ist. Auf der Liste, die *Abb. 9.7-8* zeigt, steht die Adresse des Ausgangswertes in der neuen Reihenfolge. Dabei wird nur die Adresse mit dem niedrigsten Wert angegeben. Vor Aufruf des Unterprogramms OUT1 befindet sich im Registerpaar HL die Adresse der jeweiligen Speicherstelle der Liste und in dem Register D der höherwertige Adreßteil des Speicherteils mit den Ausgangswerten.

Es wird nach Aufruf des Unterprogramms zunächst der Inhalt der Speicherzelle der Liste, dessen Adresse durch das Registerpaar HL bestimmt ist, in das Register E geschafft. Das Registerpaar DE enthält die Adresse eines Speicherplatzes, der einen Ausgangswert beinhaltet. Dieser wird mit dem Befehl

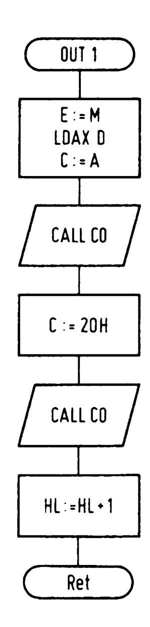


Abb. 9.7-7 Unterprogramm OUT 1

```
1150 00 08 0A 02
1154 04 0C 0E 06
1158 05 0D 0F 07
115C 01 09 0B 03
```

Abb. 9.7-8 Ausdruck der „Liste"

LDAX D in das Register A geladen. Der Inhalt des Registers A wird anschließend in das Register C geschafft und mit dem anschließenden Befehl CALL CO ausgegeben. Nun wird noch ein Leerzeichen ausgegeben. Dann wird der Inhalt des Registerpaars HL um eins erhöht, um die nächste Speicherzelle der Liste zu bestimmen.

Abb. 9.7-9 zeigt das Flußdiagramm eines Unterprogramms, das die gesamte Ausgabe des Karnaughdiagramms steuert. Dazu wird nach Start dieses Unterprogramms OUTSUB der Inhalt des Registers B mit dem Wert 04H besetzt, der Inhalt des Registerpaars HL mit dem Wert 1150H, der die Anfangsadresse der Liste darstellt. Der Inhalt des Registers D wird

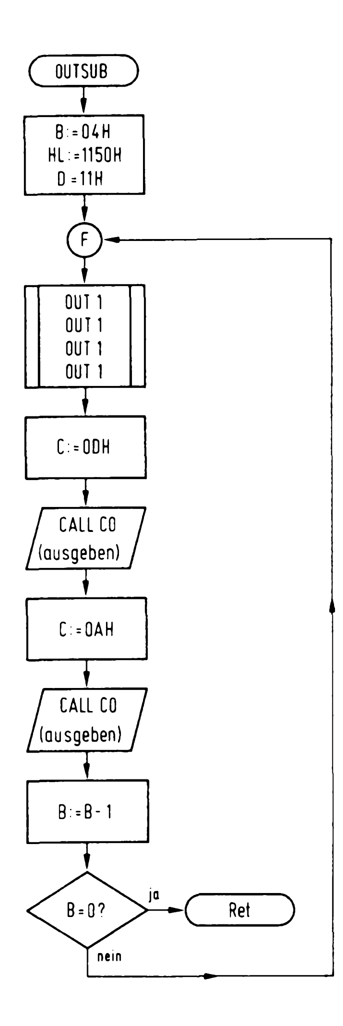


Abb. 9.7-9 Unterprogramm OUT SUB

mit dem Wert 11H besetzt. Bei der Marke F erfolgt der viermalige Aufruf des Unterprogramms OUT1 und damit die Ausgabe der ersten Zeile des Karnaughdiagramms, die ja aus vier einzelnen Ausgangswerten besteht. Anschließend wird wieder die Zeichenfolge CR, LF ausgegeben, doch diesmal wird dazu nicht der Befehl CALL ECHO verwendet, da dieser eine Veränderung des Inhalts des Registers B mit sich bringen würde. Der Inhalt des Registers B wird als nächstes um eins verringert. Ist der Wert Null geworden, so erfolgt ein Rücksprung, andernfalls erfolgt ein Sprung zur Marke F. Es wird dadurch erreicht, daß die vier Zeilen des Karnaughdiagramms ausgegeben werden.

Das eigentliche Hauptprogramm mit dem Namen MAIN zeigt *Abb. 9.7-10*. Das Programm beginnt mit der Zuweisung des Wertes 07H an das Register C. Dann wird der Inhalt

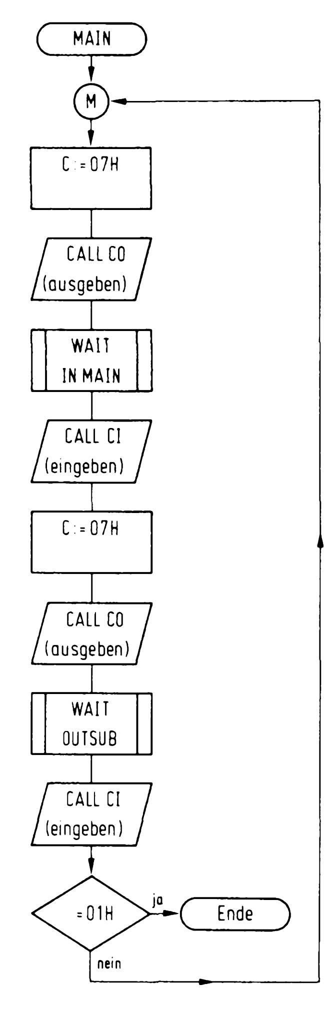


Abb. 9.7-10 Programm MAIN

mit dem Befehl CALL CO ausgegeben. Die Ausgabe dieses Befehls bewirkt ein Löschen der Anzeige auf dem Bildschirm. Es wird anschließend das Unterprogramm WAIT aufgerufen, um zu erreichen, daß so lange gewartet wird, bis der Inhalt des Bildschirms sicher gelöscht ist, bzw. bis die Abarbeitung des Befehls Clear vom Datensichtgerät erledigt wurde. Dann folgt der Aufruf des Unterprogramms IN MAIN, das für die Heranschaffung der Ausgangswerte der Funktionstabelle vom Benutzer her sorgt. Danach erfolgt mit dem Befehl CALL CI das Warten auf eine Eingabe vom Benutzer.

Gibt dieser irgendein Zeichen ein, so wird mit der Ausführung des Programms fortgefahren und der zweite Teil beginnt. Der Inhalt des Bildschirms wird wieder gelöscht, und nach dem Aufruf des Programms WAIT folgt der Aufruf des Programms OUTSUB, welches das Karnaughdiagramm ausgibt. Wieder folgt der Befehl CALL CI, nur, daß diesmal die Tasten CTRL ! betätigt wurden, falls ein Rücksprung in den Monitor erfolgt. Wurde etwas anderes eingegeben, so erfolgt ein Sprung zur Marke M und damit die Wiederholung des Programms.

Bedienungshinweise

Abb. 9.7-11 zeigt den Ausdruck des Programms im Maschinencode. Das Programm beginnt mit der Adresse 1200H. Der Start des Programms erfolgt mit der Befehlsfolge G 1280 CR.

Beim Eingeben des Programms darf die Eingabe der Liste, die Abb. 9.7-8 zeigt, natürlich nicht vergessen werden. Nach Start des Programms erfolgt die Anzeige des Codes für die vier Eingänge automatisch. Der Benutzer muß dann den Zustand 1 oder 0 entsprechend seiner Logik eingeben. Dabei ist es auch möglich, „don't cares“ mit dem Zeichen X einzugeben, wie es z. B. für Vereinfachungen von dynamischen Verknüpfungen nötig ist.

```
1200 C3 80 12 7D
1204 07 07 07 07
1208 06 04 07 DA
120C 13 12 0E 30
1210 C3 15 12 0E
1214 31 CD B0 12
1218 0E 20 CD B0
121C 12 05 C2 0A
1220 12 C9 21 00
1224 11 16 10 CD
1228 03 12 00 00
122C CD 76 12 77
1230 0E 0D CD F4
1234 01 23 15 C8
1238 C3 27 12 06
123C 0F 0E 00 CD
1240 FA 03 05 C8
1244 C3 3F 12 06
1248 04 21 50 11
124C 16 11 CD 69
1250 12 CD 69 12
1254 CD 69 12 CD
1258 69 12 0E 0D
125C CD FA 03 0E
1260 0A CD FA 03
1264 05 C8 C3 4E
1268 12 5E 1A 4F
126C CD FA 03 0E
1270 20 CD FA 03
1274 23 C9 CD FD
1278 03 4F CD FA
127C 03 C9 00 0F
1280 0E 07 CD FA
1284 03 CD 3B 12
1288 CD 22 12 CD
128C FD 03 0E 07
1290 CD FA 03 CD
1294 3B 12 CD 47
1298 12 CD FD 03
129C FE 01 CC 08
12A0 00 C3 80 12
12A4 F0 1F F0 6F
12A8 F0 2F F0 2F
12AC F0 8F F0 9F
12B0 F5 CD FA 03
12B4 F1 C9 70 4F
12B8 F0 7F F0 DF
12BC F0 57 D4 0F
```

Abb. 9.7-11 Maschinencode des Programms „Funktionstabelle Schaltalgebra“

```
1100 30 31 32 33
1104 34 35 36 37
1108 38 39 41 42
110C 43 44 45 46
```

Abb. 9.7-12 Möglicher Inhalt des Datensektors im Speicher

Abb. 9.7-12 zeigt einen Ausdruck des Speicherinhalts der Ausgangswerte. Es wurden dabei in hexadezimaler Schreibweise aufsteigend von 0 bis F die Ausgangswerte eingegeben.

9.8 Graphische Darstellung von Funktionen

9.8.1 Datensichtgerät und Mikrocomputer

Es soll ein Programm beschrieben werden, das für die Ausgabe von Daten in einer graphischen Form geeignet ist.

Es ist dabei z. B. möglich, eine Funktion in x-, y-Darstellung auszugeben, wie *Abb. 9.8.1-1* zeigt. Der auszugebende Wert wird in das Register A gegeben. Dieser Wert bestimmt die Lage des Punktes in y-Richtung. Die x-Richtung wird durch die Lage des Cursors bestimmt, der dabei am unteren Bildrand stehen muß (y = 0). Das Register A darf dabei einen Wert zwischen 0H und FH enthalten.

Programmbeschreibung

Als erstes wird das Unterprogramm PRINT nach *Abb. 9.8.1-2* besprochen, das die vorher beschriebenen Eigenschaften voll aufweist. Das Unterprogramm beginnt also zunächst mit der Anweisung B: =00H. Damit wird dem Register B der Wert 0 zugewiesen. Anschließend wird der Inhalt des Registers A, das ja den auszugebenden Wert für die y-Richtung enthält, um eins erhöht. Nun wird bei Marke 1 der Wert des Registers A wiederum um eins verringert. Es entsteht die Frage: „Warum erst erhöhen, um anschließend wieder zu erniedrigen?“

Enthält das Register A den Wert 0, so würde dies nicht erkannt werden. Denn es müssen die Zustandsbits gesetzt werden, speziell das Nullbit. Dies geschieht aber bei der Ausführung des Befehls DCR. Um den Inhalt des Registers A nicht zu verändern, ist es somit nötig, zunächst den Befehl INR A also die Erhöhung des Registers A um eins vorzuneh-

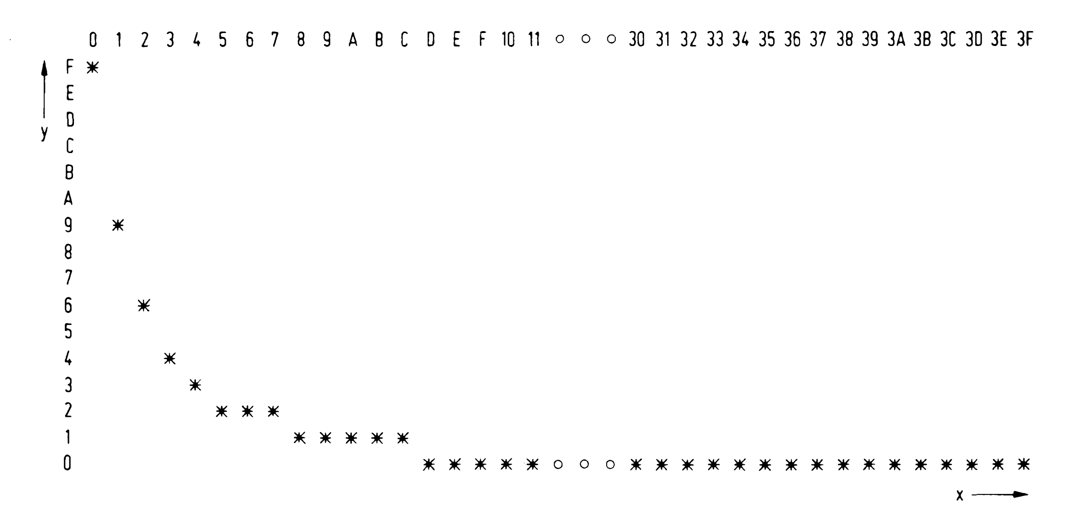


Abb. 9.8.1-1 Darstellung einer Funktionskurve auf dem Datensichtgerät

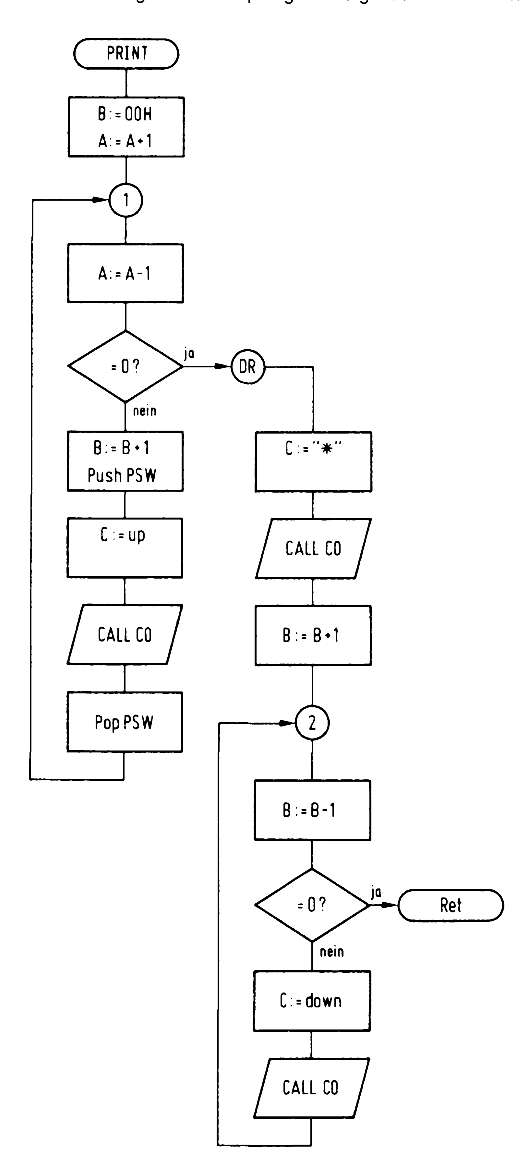


Abb. 9.8.1-2 Unterprogramm PRINT

men. Es erfolgt nach dem Befehl DCR A die Abfrage des Nullbits. Ist es gesetzt, also der Inhalt des Registers A Null, so wird zur Marke DR gesprungen und dort der Ausdruck eines Zeichens „*" bewirkt. Die genaue Erklärung dieses Vorganges erfolgt etwas später. Ist der Inhalt des Registers nicht Null, so wird zunächst einmal der Inhalt des Registers B um eins erhöht. Es enthält die Anzahl der durchlaufenen Schleifen.

Dann wird mit dem Befehl PUSH PSW der Inhalt des Registers A und der Zustandsbits in den Stapelspeicher kopiert. Dann wird dem Register C der Wert für den Befehl up zugewiesen und mit dem Befehl CALL CO ausgegeben. Anschließend wird der Inhalt des Registers A mit dem Befehl POP PSW wieder zurückgespeichert.

Es erfolgt dann ein Sprung zu der mit der Marke 1 bezeichneten Stelle. Die Schleife wird also so oft durchlaufen, bis der Inhalt des Registers A Null wird. Damit wird der Cursor um so viele Zeilen nach oben bewegt, wie das Register A angibt.

Es erfolgt der Sprung zu Marke PR.

Als erstes wird dort dem Register C der Wert für das Zeichen „*" zugewiesen, und dieses Zeichen wird mit dem Befehl CALL CO ausgegeben. Nun ist es noch nötig, den Cursor in der y-Richtung wieder an die alte Stelle zu schaffen. Dies geschieht im folgenden Programmteil. Der Inhalt des Registers B wird zunächst um den Wert 1 erhöht, um bei der Marke 2 um den Wert Eins wieder verringert zu werden. Es folgt dann eine Abfrage. Ist der Inhalt des Registers B gleich Null, so erfolgt ein Rücksprung. Andernfalls wird der Inhalt des Registers C mit dem Wert für den Befehl DOWN besetzt und anschließend wird dieser Wert mit dem Befehl CALL CO ausgegeben. Es erfolgt dann ein Sprung zur Marke 2.

Damit wäre die Beschreibung des Unterprogramms PRINT abgeschlossen. Nun soll aber auch ein Hauptprogramm zur Verfügung

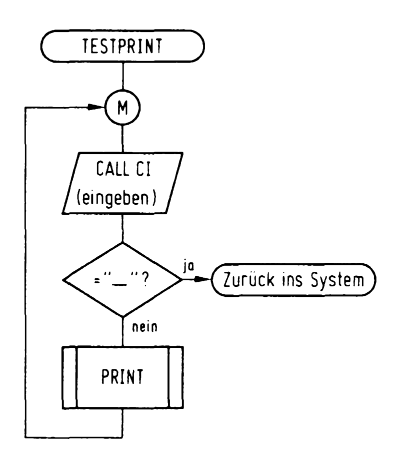


Abb. 9.8.1-3 Programm TESTPRINT

stehen, mit dem das Unterprogramm getestet werden kann. Dieses Programm zeigt *Abb. 9.8.1-3*. Das Programm hat die Aufgabe, ein Zeichen vom Benutzer eingeben zu lassen und dann das Zeichen in dieser graphischen Form auszugeben. Nach Start des Programms folgt deshalb zunächst der Befehl CALL CI. Dann wird abgefragt, ob das eingegebene Zeichen mit dem Zeichen „_“ identisch ist. Ist das Zeichen „_“ eingegeben worden, so erfolgt ein Rücksprung zum Monitorprogramm, andernfalls wird das Unterprogramm PRINT aufgerufen, und dann erfolgt ein Sprung zur Marke M, somit eine Wiederholung des Programms.

Benutzerhinweise
Abb. 9.8.1-4 zeigt das Maschinenprogramm. Das Programm wird mit der Befehlsfolge G 1322 CR gestartet. Nun kann ein Zeichen eingegeben werden.

Das Zeichen wird seinem dualen Wert entsprechend auf der y-Achse ausgegeben. Ist der duale Wert größer als 16 (dezimal), so reicht der Bildschirm nicht mehr in der Höhe aus. Ein Zeichen „*“ wird aber trotzdem ausgegeben. Ist der Wert nämlich größer als 16, so läuft der Cursor über den oberen Bildschirmrand hinaus und kommt dann von unten wieder ins Bildfeld. Durch Betätigen der Tasten CTRL space wird dem Register A beispielsweise ein Wert von 00 zugewiesen, mit CTRL ! ein Wert 01. Erfahrungen können aber am einfachsten durch eigenes Experimentieren gesammelt werden.

```
1300 06 00 3C 3D
1304 CA 12 13 04
1308 F5 0E 0B CD
130C FA 03 F1 C3
1310 03 13 0E 2A
1314 CD FA 03 04
1318 05 C8 0E 0A
131C CD FA 03 C3
1320 18 13 CD FD
1324 03 FE 5F CC
1328 08 00 CD 00
132C 13 C3 22 13
```

Abb. 9.8.1-4 Maschinencode des gesamten Programms

Dieses Unterprogramm kann recht einfach in eigenen Programmen eingesetzt werden.

Tabelle 9.8-1 zeigt das Programm noch einmal in einer anderen Darstellung. Die Entwicklung eines Programmes kann auch wie folgt erfolgen. Zuerst wird die erste Spalte mit den Adressen geschrieben, dann die dritte Spalte mit dem Befehl. Schließlich werden die Befehle codiert und in Maschinensprache in die zweite Spalte geschrieben. Die zweite Spalte kann dann in den Mikrocomputer eingegeben werden.

9.8.2 Dienstprogramm

Hier soll nun das zweite Dienstprogramm beschrieben werden. Es gestattet den Ausdruck des Programms in einem Format, wie beim Ausdruck des Programms dieses Kapitels, und es gestattet die Abspeicherung der Programme auf einem Tonbandgerät.

Tabelle 9.8-1 Programm „Graphische Darstellung von Funktionen“

Graphische Darstellung von Funktionen

```
PRINT 1300: 06   MVI B,00H
        01: 00
        02: 3C   INR A
M1      03: 3D   DCR A
        04: CA   JZ M3
        05: 12
        06: 13
        07: 04   INR B
        08: F5   PUSH PSW
        09: 0E   MVI C, „up“
        0A: 0B
        0B: CD   CALL CO
        0C: FA
        0D: 03
        0E: F1   POP PSW
        0F: C3   JMP M1
        10: 03
        11: 13
M3      12: 0E   MVI C, „*“
        13: 2A
        14: CD   CALL CO
        15: FA
        16: 03
        17: 04   INR B
```

```
M2      18: 05   DCR B
        19: C8   RZ
        1A: 0E   MVI C, „down“
        1B: 0A
        1C: CD   CALL CO
        1D: FA
        1E: 03
        1F: C3   JMP M2
        20: 18
        21: 13
TEST-   22: CD   CALL CI
PR.     23: FD
        24: 03
        25: FE   CMP „_“
        26: 5F
        27: CC   CZ „System MCS 80“
        28: 08
        29: 00
        2A: CD   CALL PRINT
        2B: 00
        2C: 13
        2D: C3   JMP TESTPR.
        2E: 22
        2F: 13
```

Programmbeschreibung

Zuerst wird das Unterprogramm OUT besprochen. *Abb. 9.8.2-1* zeigt das dazugehörige Flußdiagramm. Das Unterprogramm hat die Aufgabe, neben einem Leerzeichen den hexadezimalen Code für den Inhalt einer Speicherzelle auszugeben.

Als erstes wird der Befehl PUSH B durchgeführt. Damit wird der Inhalt der Register B und C gesichert, da das Unterprogramm sonst diese Inhalte vernichten würde. Es folgt dann die Zuweisung des Wertes 20H an das Register C und anschließend der Befehl CALL CO, wodurch ein Leerzeichen ausgegeben wird. Anschließend folgt der Befehl A: =M, wodurch der Inhalt der Speicherzelle, die durch die im Registerpaar HL enthaltene Adresse bestimmt ist, in das Register A geschafft wird. Dann folgt der Aufruf des im Monitor vorhandenen Unterprogramms NMOUT, das den Wert in Form zweier Zeichen ausgibt, die den Wert in hexadezimaler Schreibweise darstellen. Nach dieser Anweisung wird der Inhalt des Registerpaars HL um eins erhöht, um damit die nächste Speicherzelle festzulegen. An-

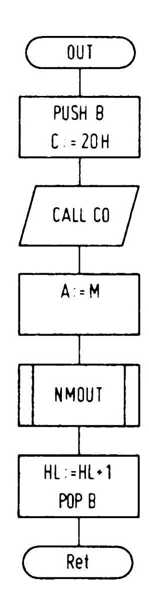


Abb. 9.8.2-1 Unterprogramm OUT

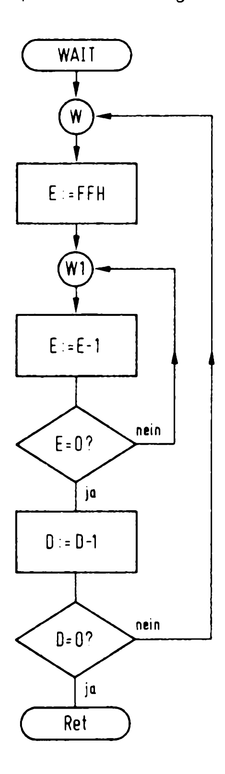


Abb. 9.8.2-2 Unterprogramm WAIT

schließend wird durch den Befehl POP B der alte Wert in das Register B und in das Register C geschafft. Erst dann folgt der Rücksprungbefehl.

Das Unterprogramm WAIT, dessen Flußdiagramm *Abb. 9.8.2-2* zeigt, hat die Aufgabe, eine Verzögerung zu bewirken. Dabei bestimmt der im Register D vor Aufruf des Unterprogramms eingegebene Wert die Verzögerungszeit.

Nach dem Start dieses Unterprogramms wird dem Register E zunächst der Wert FFH zugewiesen. Dann wird das Register E um den Wert eins verringert. Falls der sich ergebende Wert ungleich Null ist, erfolgt ein Sprung zur Marke W1, der eine Wiederholung der Verringerung bewirkt. Ist der Wert schließlich Null, so wird der Inhalt des Registers D um eins verringert. Es folgt eine Abfrage, ob der Inhalt des Registers Null geworden ist. Ist dies noch nicht der Fall, so wird zur Marke W gesprungen. Andernfalls wird das Unterprogramm durch den Rücksprungbefehl beendet.

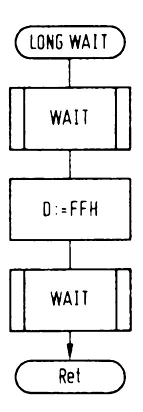


Abb. 9.8.2-3 Unterprogramm LONG WAIT

Abb. 9.8.2-3 zeigt ein Unterprogramm mit dem Namen LONG WAIT. Dieses Unterprogramm hat die Aufgabe, die maximale Verzögerungszeit des Unterprogramms WAIT etwa zu verdoppeln. Nach Aufruf des Unterprogramms LONG WAIT wird als erstes das Unterprogramm WAIT aufgerufen. Dann wird dem Register D der Wert FFH zugewiesen und erneut das Unterprogramm WAIT aufgerufen. Dann erst folgt der Rücksprungbefehl. Die zweite Zuweisung an das Register D mit dem nachfolgenden Aufruf des Unterprogramms WAIT bewirkt die Verlängerung.

Abb. 9.8.2-4 zeigt das Flußdiagramm des Hauptprogramms mit dem Namen DIENSTPROGRAMM.

Es hat die Aufgabe, zunächst zwischen dem Tonbandladeprogramm und dem Druckprogramm zu wählen. Die Entscheidung darüber kann der Benutzer fällen, außerdem ist es möglich, einen Text auszugeben. Das Programm beginnt mit dem Befehl CALL CI und wartet damit auf die Eingabe eines Zeichens vom Benutzer. Dieses Zeichen wird anschließend wieder ausgegeben mit der Befehlsfolge C: =A und CALL CO. Dann findet ein Vergleich statt. Ist das eingegebene Zeichen mit dem Code 01H identisch, so erfolgt ein Sprung zur Marke 1, andernfalls erfolgt die Abfrage auf den Code 02H. Ist das eingegebene Zeichen mit diesem Code identisch, so erfolgt ein Sprung zur Marke 2. Andernfalls wird zur Marke M gesprungen und der Programmteil wird wiederholt. Nun folgt eine Besonderheit des Programms bei Marke 1 und Marke 2. In beiden Fällen wird der Befehl RST 1 gegeben, der einen Rücksprung ins Monitorprogramm bewirkt. Bei Aufruf des Monitorprogramms als Unterprogramm wird aber die Adresse des rufenden Programms gesichert. Mit der Befehlsfolge G CR wird diese Adresse automatisch vom Monitorprogramm wieder in den Programmzähler gespeichert, und es erfolgt somit ein Rücksprung. Zweck dieses Ganzen ist die Eingabe einer Startadresse, wie sie für

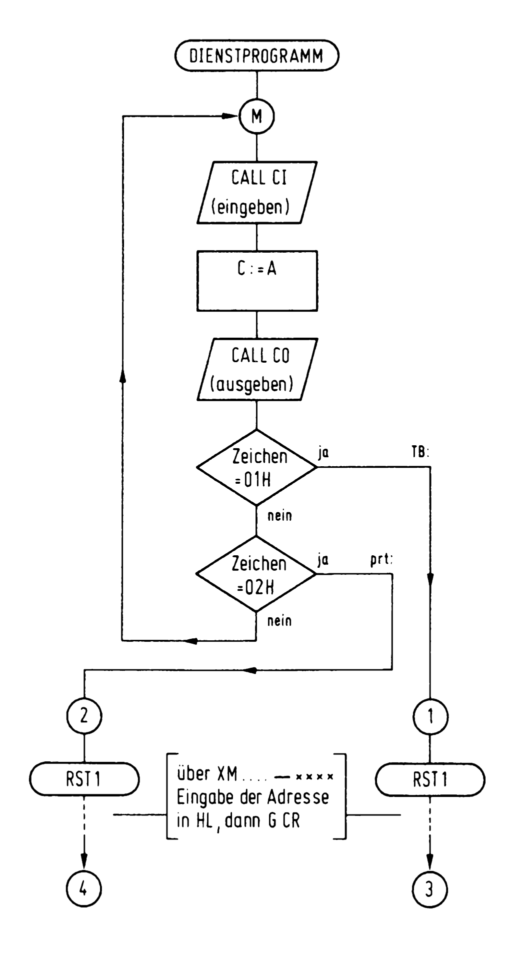


Abb. 9.8.2-4 Dienstprogramm 2, Hauptprogrammteil

das Druckprogramm als auch den Tonbandlader benötigt wird.

Mit dem Befehl XM kann die Eingabe ermöglicht werden. Nach Betätigen der Tastenfolge XM wird der alte Inhalt des Registerpaars HL ausgegeben und zwar so, wie er für eine Adresse verwendet wird. Danach ist es möglich, diesen Inhalt zu ändern. Es wird einfach die neue Adresse eingegeben und danach CR betätigt. Dann wird G CR eingegeben, um einen Rücksprung in das Hauptprogramm zu gestatten. Der Wert in HL, der vorher eingegeben wird, befindet sich nun auch wirklich in

dem Registerpaar HL. Zum Zeitpunkt der Eingabe mit Hilfe des Monitors wird er auf einen Stapelspeicher geschafft. Nun ist es dem Programm möglich, diesen Wert weiterzuverarbeiten. Erfolgte der RST-1-Befehl von der Marke 1 aus, so wird nun der Programmteil, der bei der Marke 3 beginnt, abgearbeitet (*Abb. 9.8.2-5*). Wurde der RST-1-Befehl bei der Marke 2 gegeben, wird der Programmteil bei der Marke 4 abgearbeitet (*Abb. 9.8.2-6*).

Als nächstes wird der Programmteil mit der Marke 2 besprochen. Er stellt das Tonbandladeprogrammteil dar und hat die Aufgabe, den Inhalt des Programmspeichers von der angegebenen Startadresse bis zur nächsten $^1/_4$-K-Grenze in hexadezimaler Schreibweise auszugeben. Ist beispielsweise als Startadresse 1234H angegeben worden, so wird der Inhalt des Programmspeichers bis zur Adresse 1300H ausgegeben.

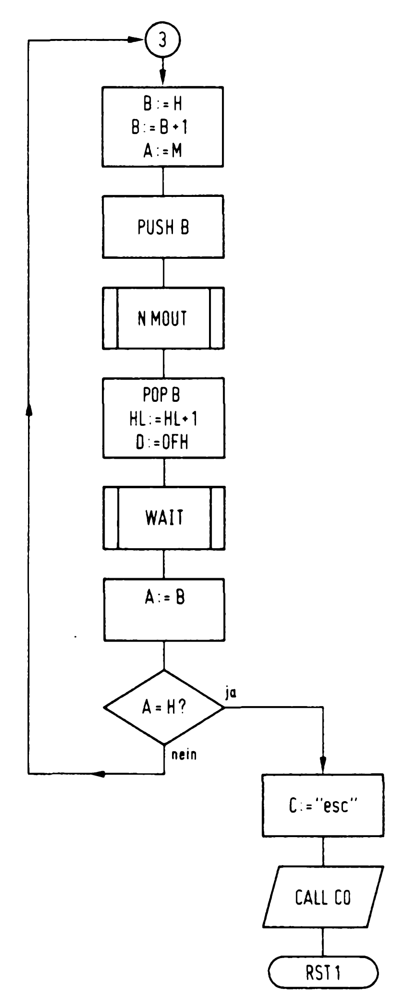


Abb. 9.8.2-5 Programmteil 3

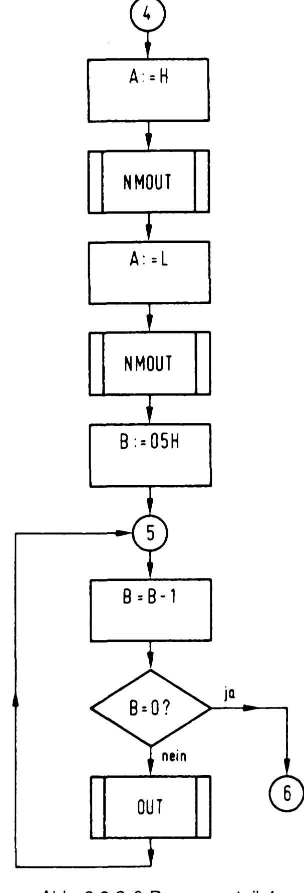


Abb. 9.8.2-6 Programmteil 4

Das Programm beginnt mit der Kopierung des Inhalts des Registers H in das Register B. Dann wird der Inhalt des Registers B um eins erhöht. Dadurch wird die nächste $^1/_4$-K-Grenze festgelegt. Bei dem hier angegebenen Tonbandlade-Programmverfahren kann maximal $^1/_4$ K Programmspeicherinhalt in einem Zuge auf Tonband abgespeichert werden, wie später noch genauer erklärt wird.

Nun wird der Inhalt der Speicherzelle, die durch den Inhalt des Registerpaars HL festgelegt ist, in das Register A geschafft. Dann werden durch den Befehl PUSH B die Inhalte der Register B und C auf den Stapelspeicher gebracht. Anschließend wird das Unterprogramm NMOUT aufgerufen, das sich im Monitor befindet, und – wie schon gesagt – die Aufgabe hat, den Inhalt des Registers A in zwei Zeichen in hexadezimaler Schreibweise auszugeben. Nun wird der Inhalt der Register B und C mit dem Befehl POP B wieder zurückgespeichert und es erfolgt eine Erhöhung des Inhalts des Registerpaars HL. Dadurch wird die nächste Speicherzelle festgelegt. Nun folgt die Zuweisung des Wertes 0FH an das Register D und anschließend der Aufruf des Unterprogramms WAIT. Dadurch wird eine geringe Verzögerung erreicht, die sich günstig auf die Störsicherheit beim Aufzeichnen der Daten auf das Tonband auswirkt. Nun wird der Inhalt des Registers B in das Register A kopiert und dieser Inhalt kann dann mit dem Inhalt des Registers H verglichen werden. Sind die beiden Inhalte nicht gleich, so erfolgt ein Sprung zur Marke 3, weil die $^1/_4$-K-Grenze noch nicht erreicht wurde. Andernfalls wird in das Register C der Wert für das Zeichen „esc" geschafft und mit dem Befehl CALL CO ausgegeben. Dadurch wird später beim Wiedereinlesen des Programms vom Tonband das Ende des Programmspeicherinhaltes vom Monitorprogramm erkannt. Als nächster Befehl folgt RST 1, um wieder in das Monitorprogramm zurückzukehren. Nun zu dem etwas umfangreicheren Programmteil, der bei der Marke 4 beginnt. Er hat die Aufgabe, den Inhalt des Programmspeichers auf einem Drucker auszugeben. Dabei ist wieder die umgekehrte Ausgabe für den hier verwendeten Drucker berücksichtigt. Auch hier besteht die Möglichkeit für den Anwender, diese zu ändern, wenn es gewünscht wird.

Das Programm beginnt mit der Kopierung des Inhalts des Registers H in das Register A. Dieser Wert wird anschließend mit dem Befehl CALL NMOUT in hexadezimaler Schreibweise ausgegeben. Auf die gleiche Weise wird der Inhalt des Registers L ausgegeben. Nun wird das Register B mit dem Wert 05H besetzt. Anschließend wird bei der Marke 5 eins von diesem Wert abgezogen. Ist der Wert gleich Null, so erfolgt ein Sprung zur Marke 6. Andernfalls wird das Unterprogramm OUT aufgerufen. Es hat die Aufgabe, neben einem Leerzeichen, das zuerst ausgegeben wird, den Inhalt der Speicherzelle auszugeben, die durch den Inhalt des Registerpaars HL bestimmt ist. Auch die Erhöhung des Inhalts des Registerpaars HL um eins wird vorgenommen. Danach folgt der Sprung zur Marke 5 und damit eine Wiederholung. Die Schleife und somit der Aufruf des Unterprogramms wird insgesamt viermal durchgeführt. Damit werden vier aufeinanderfolgende Speicherzellen ausgegeben.

Es folgt der Sprung zur Marke 6 (*Abb. 9.8.2-7*). Der Programmteil von Marke 6 bis zu Marke 8 dient dazu, den Inhalt des Registerpaars HL um 8 zu verringern, so daß der „umgekehrte" Ausdruck ermöglicht wird. Ist ein Drucker vorhanden, der „normal" ausdruckt, also das Papier von unten nach oben transportiert, kann dieser Programmteil weggelassen werden.

Es folgt die Beschreibung. Dem Register B wird als erstes der Wert 09H zugewiesen. Dann folgt bei der Marke 7 die Verringerung des Inhalts des Registers B um eins und anschließend wird gefragt, ob dieser Inhalt den Wert 0 besitzt. Ist dies nicht der Fall, also der Inhalt des Registers noch ungleich 0, so wird

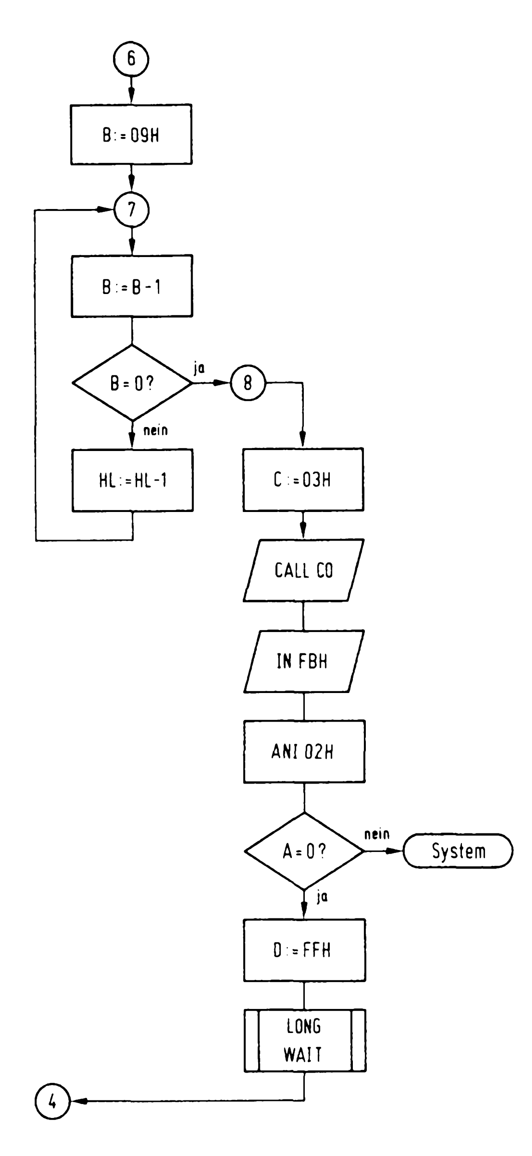


Abb. 9.8.2-7 Programmteil 6

der Inhalt des Registerpaars HL um eins verringert. Andernfalls erfolgt der Sprung zu Marke 8.

Dort wird dem Register C der Wert 03H zugewiesen, der den Druckbefehl auslöst. Dieser wird dazu mit dem Befehl CALL CO ausgegeben. Nun folgt der Befehl IN FBH. Damit wird der Inhalt eines Statusregisters im UART des Mikrocomputersystems herbeigeschafft und im Register B abgelegt. Durch den Befehl ANI 02H werden die nicht interessierenden Informationsbits dieser Statusinformation ausgeblendet. Zurück bleibt aufgrund der durchgeführten UND-Verknüpfung mit dem Wert 02H dann nur das zweite Bit der Information. Ist dieses Bit ungleich Null, so erfolgt der Sprung zum Monitorprogramm. Andernfalls wird dem Register D der Wert FFH zugewiesen, und es erfolgt dann der Aufruf des Unterprogramms LONG WAIT. Damit wird sichergestellt, daß der Druckvorgang beendet wurde, bevor der Sprung zur Marke 4 und damit eine Wiederholung des Programmteils erfolgt.

Benutzerhinweise

Abb. 9.8.2-8 zeigt den Maschinencode für das gesamte Programm. Der Start des Programms erfolgt dabei durch die Befehlsfolge G 1218 CR. Nun ist es möglich, einen beliebigen Text zu schreiben. Wird die Tastenfolge CTRL ! betätigt, so erfolgt die Auswahl des Tonbandladers. Erfolgt die Betätigung der Tasten CTRL ", so erfolgt die Auswahl des Druckprogramms.

Wird also das Druckprogramm gewünscht, so kann man nach Betätigung der Tasten CTRL ", die ein Aufruf des Monitors bewirkt, mit dem Befehl XM eine Eingabe der Startadresse vornehmen, genauer gesagt, der Endadresse, wegen des Rückwärtsausdrucks. Der Druckvorgang wird dann einfach durch die Befehlsfolge G CR ausgelöst. Ist man bei der eigentlichen Anfangsadresse angelangt, so kann man dies auf dem Datensichtgerät verfolgen. Man betätigt einfach irgendeine Taste und nach Beenden der letzten Druckzeile erfolgt dann ein Sprung in das Monitorprogramm. Der Drucker besitzt eine Leitung zum Auslösen des Druckvorgangs, diese kann man an den Ausgang 03H des Befehlsdecodierers des Datensichtgerätes anschließen. Dadurch ist es

```
1200 1E FF 1D C2
1204 02 12 15 C2
1208 00 12 C9 C5
120C 0E 20 CD FA
1210 03 7E CD C3
1214 02 23 C1 C9
1218 CD FD 03 4F
121C CD FA 03 FE
1220 01 CA 5F 12
1224 FE 02 CA 2C
1228 12 C3 18 12
122C CF 7C CD C3
1230 02 7D CD C3
1234 02 06 05 05
1238 CA 41 12 CD
123C 0B 12 C3 37
1240 12 06 09 05
1244 CA 4B 12 2B
1248 C3 43 12 0E
124C 03 CD FA 03
1250 DB FB E6 02
1254 C4 08 00 16
1258 FF CD 79 12
125C C3 2D 12 CF
1260 44 04 7E C5
1264 CD C3 02 C1
1268 23 16 0F CD
126C 00 12 78 BC
1270 C2 60 12 0E
1274 1B CD FA 03
1278 CF CD 00 12
127C 16 FF CD 00
1280 12 C9 00 F3
```

Abb. 9.8.2-8 Maschinencode des Dienstprogramms 2

dem Programm möglich, den Druckvorgang automatisch auszulösen. Nun zu dem Tonbandladeprogramm.

Dazu wird das Tonbandgerät eingeschaltet und auf „Aufnahme“ gestellt. Der Eingang (umschaltbarer Ein-/Ausgang) des Tonbandspeichers liegt am Ausgang des Mikrocomputers und wird parallel hinzugeschaltet. Zuvor wurde durch die Befehlsfolge G 1218 CR das Dienstprogramm gestartet. Während der Aufnahme kann ein Text eingegeben werden. Zweckmäßigerweise wird mit der Eingabe eines Clear-Befehls für das Datensichtgerät begonnen. Dies geschieht durch die Tastenfolge CTRL'. Anschließend kann ein Text eingegeben werden, der angibt, um was für ein Programm es sich handelt, wo es gestartet wird usw. Dann wird wieder ein Clear-Befehl ausgegeben und die Tasten CTRL ! werden betätigt. Es meldet sich der Monitor. Als nächstes wird der Befehl XM an den Monitor gegeben, worauf dieser den im Register HL enthaltenen Wert ausgibt. Dann wird der Wert oder die Startadresse angegeben. Am zweckmäßigsten wird der Beginn eines $^1/_4$-K-Blocks genannt, auf dem das Programm vorkommt oder beginnt, also zum Beispiel 1200H. Werden jetzt die Tasten G und CR betätigt, so erfolgt die Ausgabe des Speicherinhalts. Nachdem diese beendet ist, kann das Tonbandgerät nach kurzer Zeit abgeschaltet werden. Die Aufzeichnung ist damit beendet. Ist das Programm noch nicht ganz ausgelesen, so wird der Vorgang wiederholt, so lange, bis alles Gewünschte auf Band ist. Die Rückspeicherung vom Tonband geschieht wie folgt. Der Ausgang des Tonbandspeichers liegt an dem Eingang des Datensichtgerätes, also parallel zum Ausgang des Mikrocomputers. Dazu wird am besten, wie schon gesagt, ein im Schaltplan des Tonbandspeichers eingezeichneter umschaltbarer BUS-Aus-/Eingang verwendet.

Nun wird das Tonbandgerät bei der Betriebsart „Wiedergabe“ eingeschaltet. Es folgen dann der Reihe nach: Die Löschung des Bildschirms, die Ausgabe des Textes, anschließend eine erneute Löschung des Bildschirms, sowie der Ausdruck über die Vorgänge beim Arbeiten mit dem Monitorprogramm und schließlich der Inhalt des Programmspeichers. Der gesamte Bildschirm ist nun knapp vollgeschrieben. Dadurch ergibt sich die Einteilung in $^1/_4$ Blöcke. Wäre nämlich ein $^1/_2$ Block ausgegeben worden, so wäre kein Platz mehr auf dem Bildschirm für zusätzliche Eingaben geblieben, wie sie gleich benötigt werden.

Der Inhalt des Programmspeichers steht nun zwar auf dem Bildschirm, er ist aber noch nicht im Programmspeicher des Mikrocomputers. Dazu wird wie folgt weiterverfahren. Zunächst wird der Cursor an die untere Stelle des aufgezeichneten Speicherinhalts gelegt.

Dann werden die Befehle I xxxx CR ausgegeben, wobei für xxxx die Adresse eingesetzt werden muß, von der ab das Programm in den Speicher eingelesen werden soll.

Nun wird das Datensichtgerät auf die Betriebsart „read" mit der Taste „read" geschaltet. Dann wird der Cursor auf den ersten ausgegebenen Speicherzellen-Inhalt positioniert und die Leertaste betätigt. Danach wird der gesamte Speicherinhalt in den Speicher des Mikrocomputers eingelesen, bis die „Ende"-Marke erreicht ist, die automatisch ausgegeben wurde. Dann springt der Cursor willkürlich. Der Einlesevorgang wird durch Betätigen der Taste „write" endgültig beendet. Sollte einmal nach Betätigen der Taste „space", also der Leertaste, der Einlesevorgang nicht korrekt erfolgen, so kann ein fehlerhaftes Zeichen vorliegen. In diesem Fall muß der Einlesevorgang mit dem Tonbandgerät wiederholt werden. Das fehlerhafte Zeichen kann aber auch korrigiert werden. Auch kann ein vorübergehender Systemfehler vorliegen, der im Datensichtgerät liegt. Dann muß die Befehlsfolge I xxxx CR und die Positionierung mit nachfolgendem Start wiederholt werden. Dabei ist zu beachten, daß beim Umschalten von „read" auf „write" Bildschirmteile durch den Monitor gelöscht werden können. Diese Umschaltung muß aber vorgenommen werden, um eine neue Eingabe zu ermöglichen.

9.9 Analoge Darstellung von digital gespeicherten Funktionen

9.9.1 Datensichtgerät und D/A-Umsetzer

Hier soll zur Abwechslung nun wieder ein etwas mehr auf Hardware orientiertes Kapitel eingeschoben werden.

Es soll besprochen werden, wie es mit dem beschriebenen Datensichtgerät und einem D/A-Umsetzer möglich ist, beliebige Funktionen mit einer Auflösung von 6 oder 8 bit in der y-Richtung und von maximal 1024 bzw. 2048 Punkten in der x-Richtung, analog auszugeben. Dabei wird der Bildschirmspeicher als eigentlicher Funktionsspeicher eingesetzt. Die Speicherung geschieht in dualer Form. Die Eingabe der Form erfolgt im einfachsten Fall durch die Tastatur. Dabei können aber nur 6 bit eingegeben werden. Um aber 8 bit zu erreichen, ist es nötig, Veränderungen an der Schaltung des Datensichtgerätes vorzunehmen. Der Befehlsdecodierteil des Datensichtgerätes, wie schon in der Beschreibung des Datensichtgerätes erwähnt, muß diesmal ausgeschaltet werden.

Der D/A-Umsetzer, für den am besten eine integrierte Version verwendet wird, wird an den Peripherieausgang, gegebenenfalls über eine Zwischenspeicherstufe, angeschlossen. Die Übernahme in den Zwischenspeicher erfolgt dann durch das TK-Signal für den Peripherieausgang. Der serielle Ausgang des Datensichtgerätes wird mit dem seriellen Eingang des Datensichtgerätes verbunden. Es folgt die Eingabe der auszugebenden Funktion. Ist dies erfolgt, so wird das Datensichtgerät auf die Betriebsart „read" umgeschaltet. Die Ausgabe der Funktion erfolgt z. B. nach Betätigen der Leertaste. Die Funktion wird am Ausgang des D/A-Umsetzers in analoger Darstellung mit einer Geschwindigkeit ausgegeben, die durch die Übertragungsgeschwindigkeit des UART bestimmt ist. Damit ist eine Veränderung der Ausgangsfrequenz durch Veränderung der Taktfrequenz des UART möglich.

Abb. 9.9.1-1 zeigt die schematische Zusammenschaltung von D/A-Umsetzer und Datensichtgerät.

Nun noch etwas zu Eingabe der Funktion: Angenommen, die Funktion wird mit Hilfe der

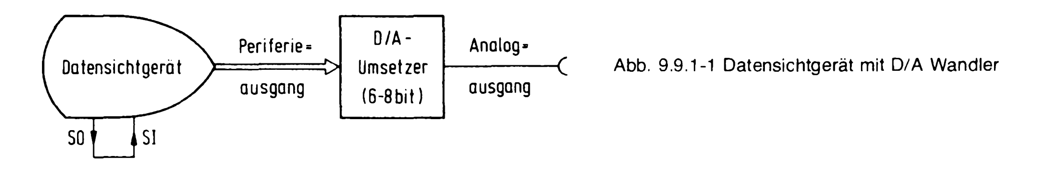


Abb. 9.9.1-1 Datensichtgerät mit D/A Wandler

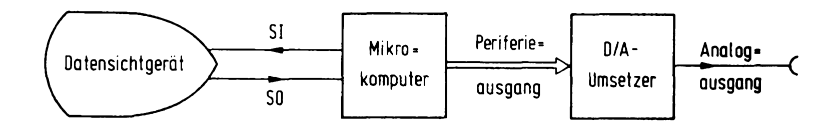


Abb. 9.9.2-1 Vollständiges System für analoge Ausgabe

Tastatur eingegeben, und außerdem werden nur die 6 bit verarbeitet. Der D/A-Umsetzer verarbeitet die eingegebenen Zeichen später entsprechend ihrem dualen Wert. Soll die auszugebende Funktion ihren tiefsten Punkt in der y-Richtung erreichen, so ist der Code 00H in den Speicher einzutragen. Da aber ein Steuerzeichen nicht eingespeichert wird, ohne daß die besagten Veränderungen vorgenommen sind, und da aber die ersten beiden höherwertigen Stellen ohne Bedeutung für die auszugebende Funktion bleiben (6 bit), so ist es auch möglich, den Code 40H einzugeben. Dies entspricht aber der Taste à. Um den höchsten Punkt der Funktion in y-Richtung auszugeben, ist die Einspeicherung des Codes 3F nötig. Dies geschieht durch Eingabe des Zeichens „?“. Hier liegt ein Sprung vor, da der Code 40H größer ist als der Code 3FH. Der Sprung erfolgt bei dem Code 5F durch das Zeichen „–“, und dem nächsten Code, der eingegeben werden kann, nämlich 20H, welcher durch das Zeichen „space“ realisierbar ist. Der Sprung wurde hier vorgesehen, weil es möglich sein kann, daß eine Tastatur vorhanden ist, die keine Kleinbuchstaben ausgeben kann. So aber ist es möglich, auf jeden Fall auch nur mit der Genauigkeit von 6 bit alle Funktionswerte mit der einfacheren Tastatur einzugeben.

9.9.2 Datensichtgerät, Mikrocomputer und D/A-Umsetzer

Abb. 9.9.2-1 zeigt die schematische Anordnung dieses umfangreichen Systems. Es besteht jetzt die Möglichkeit, die Funktion mit dem Datensichtgerät in irgendeiner Form bequem einzugeben und im Arbeitsspeicher des Mikrocomputers abzulegen. Die Funktion könnte dann beispielsweise mit dem im vorherigen Kapitel angegebenen Programm zur Kontrolle noch einmal in graphischer Form ausgegeben werden, um dann über den D/A-Umsetzer in analoger Form aufgezeichnet zu werden. Dabei kann der Ablauf auf einfache Weise vom Mikrocomputer kontrolliert und gesteuert werden. Der Mikrocomputer hätte dann die Aufgabe, die Funktion dual über einen Peripherieausgang mit einer gewünschten Geschwindigkeit auszugeben. Dabei ist die Frequenz natürlich durch die maximalen Ausführungszeiten des Mikrocomputers begrenzt.

9.10 Erkennen von Wörtern einer Programmiersprache

Dies ist ein interessantes Thema. Die Erkennung von Wörtern spielt nämlich in der Computertechnik eine große Rolle. Auch in den

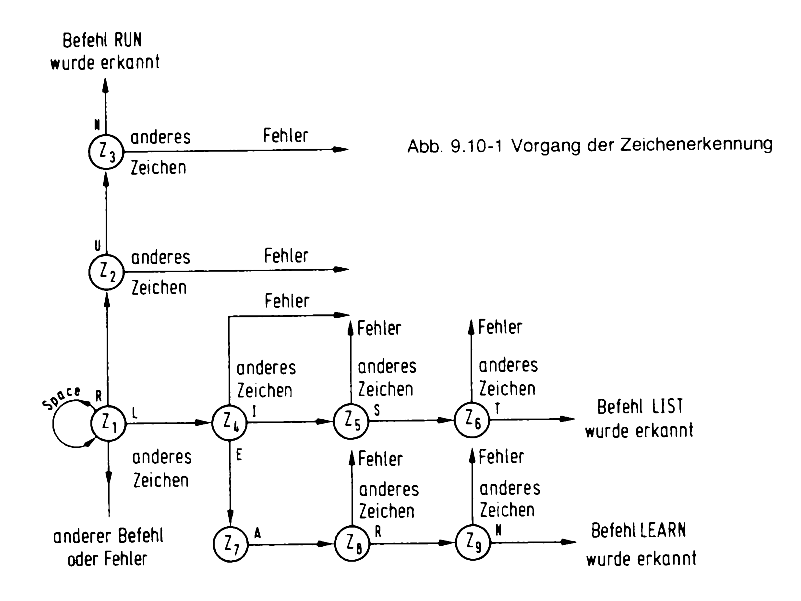


Abb. 9.10-1 Vorgang der Zeichenerkennung

vorherigen Kapiteln wurde teilweise vom Computer ein solcher Erkennvorgang durchgeführt. Nur waren es dort keine Wörter, die eingegeben wurden, sondern meist nur einzelne Zeichen.

Abb. 9.10-1 zeigt ein Beispiel eines Erkennvorgangs in einer Graphendarstellung. Diese Darstellungsweise wurde bisher noch nicht verwendet. Sie ähnelt einem Flußdiagramm, nur daß zum Beispiel die einzelnen Maschinenbefehle nicht eingezeichnet sind, und bei den einzelnen Verknüpfungspunkten mehrere Entscheidungen möglich sind. In dieser Darstellungsart ist es besonders einfach, solche Erkennungsvorgänge darzustellen.

Nun zur Erklärung der Abbildung: Es wird von dem Zustand Z 1 ausgegangen. Dies sei der Anfangszustand in der Maschine. Wird ein Zeichen „space" eingegeben, so bleibt die Maschine in diesem Zustand Z 1. Wird ein Zeichen „R" eingegeben, so gelangt die Maschine in den Zustand Z 2. Wird nun ein Zeichen eingegeben, das nicht dem Zeichen „U" entspricht, so wird ein Fehler erkannt. Wird aber das Zeichen „U" eingegeben, so gelangt die Maschine in den Zustand Z 3. Nun muß das Zeichen „N" eingegeben werden. Wird dies nicht getan, so erfolgt die Erkennung eines Fehlers. Andernfalls wäre die Zeichenfolge „RUN" erkannt worden. Nun zurück zum Zustand Z 1. Wird das Zeichen „L" eingegeben, so gelangt die Maschine in den Zustand Z 4. Wird nun ein anderes Zeichen als „I" oder „E" eingegeben, so erfolgt die Erkennung eines Fehlers, wird „I" eingegeben, so gelangt die Maschine in den Zustand Z 5 und wird „E" eingegeben, so gelangt sie in den Zustand Z 7. Es erfolgt nun in beiden Fällen ein weiterer Erkennungsvorgang, und es wird entweder ein Fehler erkannt, oder die Zeichenfolge „LIST" oder die Zeichenfolge „LEARN".

Abb. 9.10-2 zeigt ein dazugehöriges Flußdiagramm. Es ist deutlich zu erkennen, daß durch die Graphendarstellung in Abb. 9.10-1 eine erheblich größere Übersichtlichkeit gewonnen wird. Zum Erstellen eines Programms

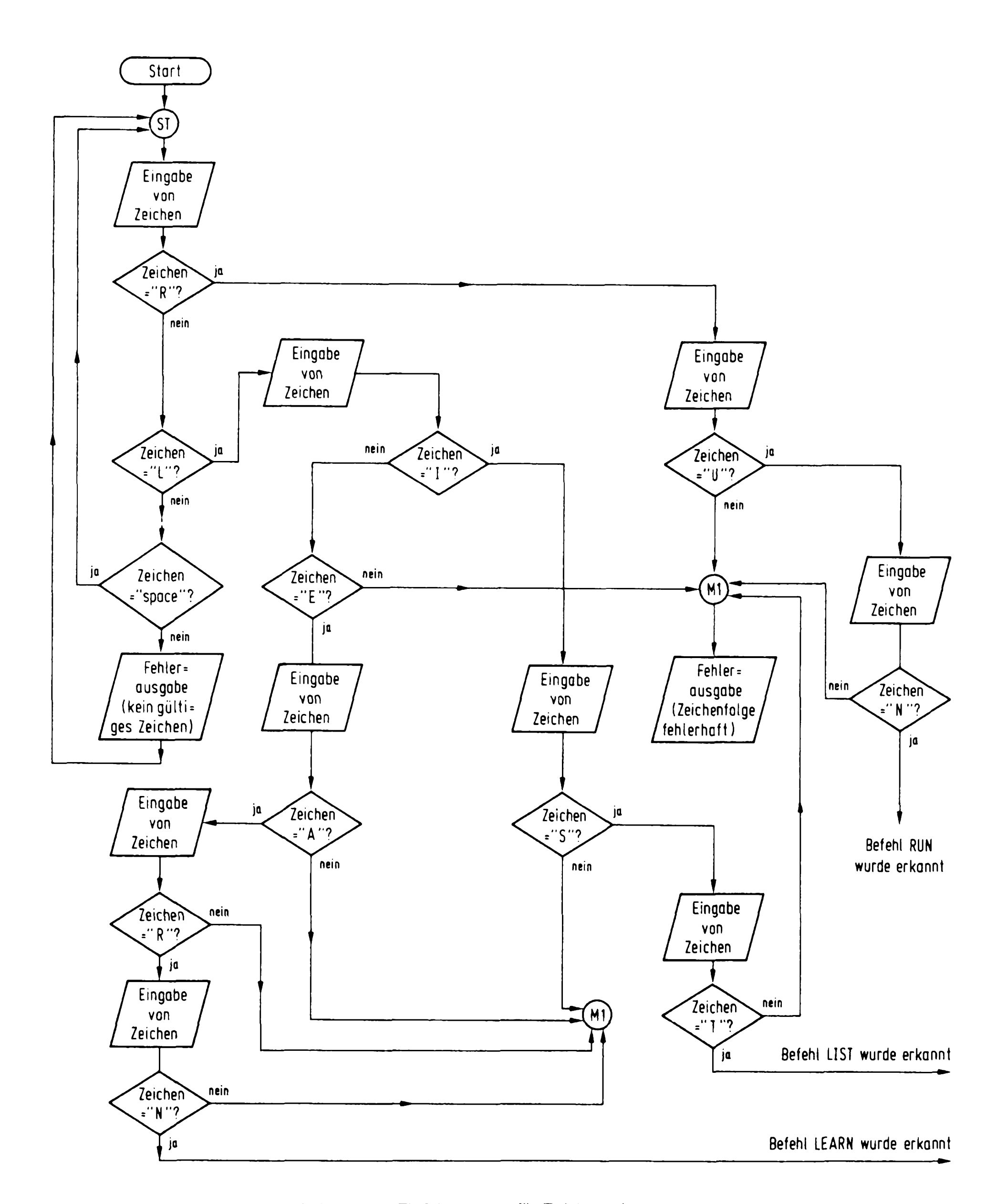


Abb. 9.10-2 Flußdiagramm für Zeichenerkennung

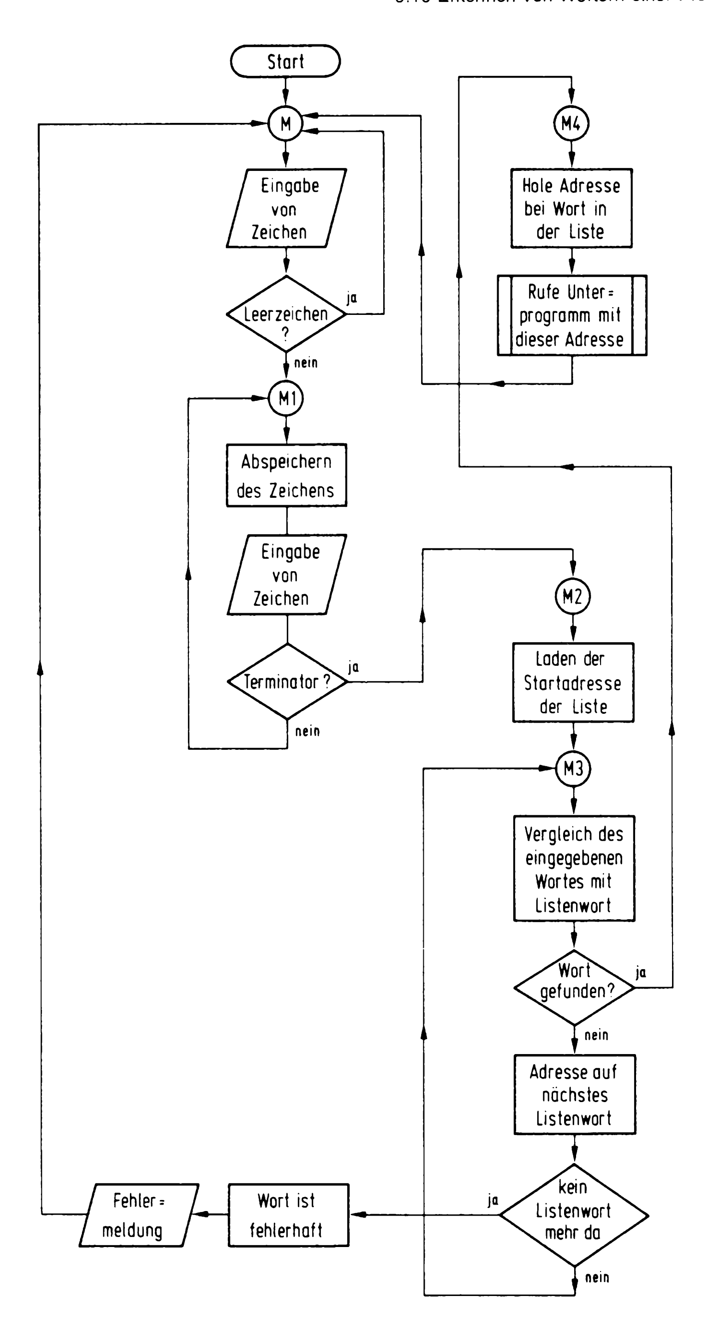


Abb. 9.10-3 Zeichenerkennung mit Hilfe einer Liste

ist aber der Schritt über ein Flußdiagramm von Vorteil. Es gibt noch eine andere Möglichkeit, Wörter zu erkennen. Dazu wird das Wort dem Computer eingegeben, nur wird dieses Wort im Speicher des Computers zunächst abgespeichert. Dann besitzt der Computer noch eine Liste mit Wörtern, die zugelassen sind, und bei jedem Wort steht auch noch die Adresse mit dem dazugehörigen Bearbeitungsprogramm. Nun vergleicht der Computer schrittweise das eingegebene Wort mit den Wörtern, die in der Liste vorhanden sind. Ist das Wort in der Liste gefunden worden, so wird die beim Wort in der Liste stehende Adresse verwendet. Mit dieser Adresse kann zum Beispiel ein Unterprogramm gestartet werden. Wird das Wort in der Liste nicht gefunden, so erfolgt die Ausgabe einer Fehlermeldung. Dieses zweite Verfahren hat den Vorteil gegenüber dem zuerst beschriebenen, daß auch Ergänzungen des „Wortschatzes" möglich sind, in dem einfach die Liste durch dieses Wort erweitert wird. Im ersten Fall wäre nicht nur eine Änderung eines Datenbereichs erforderlich, wie hier, sondern eine vollständige Programmänderung.

Abb. 9.10-3 zeigt das Flußdiagramm für eine solche Erkennung. Nach dem Start des Programms erfolgt die Eingabe eines Zeichens. Wurde das Leerzeichen eingegeben, so erfolgt eine Wiederholung des Eingabevorgangs. Wird ein anderes Zeichen als das Leerzeichen eingegeben, so erfolgt der Sprung zur Marke M 1. Dort wird dieses Zeichen auf einen Hilfsbereich abgespeichert. Dann erfolgt die erneute Eingabe eines Zeichens. Die Erkennung vom Ende eines Wortes erfolgt durch einen sogenannten Terminator. Man kann zum Beispiel CR als Terminator definieren. Wird kein Terminator eingegeben, so erfolgt ein Sprung zur Marke M 1 und damit eine Wiederholung des Abspeicher- und Eingabevorgangs. Wird das Wort durch den Terminator beendet, so erfolgt ein Sprung zur Marke M 2.

Es erfolgt jetzt das Laden der Startadresse

Liste:

Adresse des Listenworts	Wort	Adresse für Unterprogramm
00	RUN	100
10	LIST	200
20	LEARN	300
	⋮	

Abb. 9.10-4 Liste für die Zeichenerkennung

der Liste. Es folgt dann der Vergleich des eingegebenen Wortes mit dem Wort in der Liste. *Abb. 9.10-4* zeigt das Beispiel einer Liste, um deren Aufbau zu verdeutlichen. Der Vergleich kann beispielsweise mit einem Unterprogramm erfolgen. Ist das Wort gefunden, so folgt ein Sprung zur Marke M 4. Andernfalls wird die Adresse des nächsten Listenwortes bestimmt. Ist kein Listenwort mehr vorhanden, was zum Beispiel durch Vergleich der Adresse des Listenwortes mit einer vorgegebenen Maximaladresse geschehen kann, so erfolgt die Ausgabe einer Fehlermeldung, denn dann wurde das eingegebene Wort nicht erkannt. Andernfalls folgt der Sprung zur Marke M 3 und damit ein neuer Vergleich. Wurde nun aber ein Wort erkannt, so erfolgt der Sprung zur Marke M 4. Dort wird als nächstes eine in der Liste vorhandene Adresse herbeigeschafft, die beispielsweise bei dem erkannten Wort steht. Es folgt dann beispielsweise der Aufruf eines Unterprogramms mit dieser Adresse. Damit wird dem eingegebenen Wort gemäß eine Handlung ausgeführt. Es folgt danach ein Sprung zur Marke M und damit eine Wiederholung des Eingabevorgangs. Diese Programme sollten nur ein Beispiel von vielen darstellen und der Verdeutlichung der Arbeitsweise von solchen Erkennungsvorgängen dienen.

9.11 Erkennung von falsch geschriebenen Wörtern und deren automatische Korrektur

Hier soll nun kurz ein Gebiet angesprochen werden, auf dem noch viel geforscht werden kann. Die Erkennung von falsch geschriebenen Wörtern hat schon viele Programmierer beschäftigt. Es gibt dabei die unterschiedlichsten Methoden mit unterschiedlicher Effizienz.

Ein Beispiel wäre, vom Computer her alle Möglichkeiten entsprechend dem ersten Beispiel des vorhergehenden Kapitels durchzuprobieren. Dies würde aber einen enormen Aufwand bedeuten, der nicht tragbar ist. Eine wesentliche Vereinfachung ergibt sich, wenn man ein Programm mit der Struktur des zweiten Beispiels des vorhergehenden Kapitels verwendet. Es muß nun allerdings ein Zusatzprogramm geschaffen werden, das die Ähnlichkeit von Wörtern mit einer bestimmten Zahl festhält. Je ähnlicher ein Wort zu dem anderen ist, eine um so größere Zahl wird zugewiesen. Nun wird das eingegebene Wort mit allen Wörtern der Liste verglichen. Jedem Wort der Liste wird eine Zahl, die den Grad der Ähnlichkeit zu dem eingegebenen Wort entspricht, beispielsweise auf einer gesonderten Liste zugewiesen. Dann wird *das* Wort der Liste ausgewählt, welches die größte Zahl erhielt. Dieses Wort kann dann beispielsweise an den Benutzer ausgegeben werden, um eine Kontrolle zu ermöglichen. Das ganze Problem liegt in dem Programm, das die Ähnlichkeit bestimmen soll. Es können z. B. die Anzahl der übereinstimmenden Buchstaben als Zahl gewählt werden, wobei die Folge dieser Buchstaben mit berücksichtigt werden muß. Wird z. B. ein Wort „LEST" eingegeben. Ein Vergleich mit dem Wort „RUN" ergibt die Zahl 0. Ein Vergleich mit dem Wort „LIST" aber ergibt die Zahl 3, da drei Buchstaben übereinstimmen. Ein Vergleich mit dem Wort „LEARN" ergibt den Wert 2. Das Wort LIST besitzt die größte Ähnlichkeitszahl und wird somit erkannt. Schwierigkeiten gibt es aber, wenn man Buchstaben entsprechend ihrer Reihenfolge vergleicht. Also ersten Buchstaben des eingegebenen Wortes mit erstem Buchstaben des Listenwortes und dann zweiten Buchstaben des eingegebenen Wortes mit zweitem Buchstaben des Listenwortes usw. Wird nämlich z. B. ein Buchstabe bei der Eingabe ausgelassen oder hinzugefügt, so funktioniert das angegebene Zählverfahren nicht mehr. Ein Beispiel: Eingegeben wurde „LST". RUN erhält die Zahl 0, LIST die Zahl 1 und LEARN die Zahl 1. Hier ist eine Entscheidung nicht mehr möglich. Es muß bei der Gewinnung der Ähnlichkeitszahl also auch das Auslassen oder Hinzufügen einzelner Buchstaben berücksichtigt werden. Dies kann eventuell durch Probieren aller Möglichkeiten der Anordnung geschehen, indem der Computer mehrere solcher Zähldurchgänge vornimmt und dann jeweils selbst Buchstaben, am besten Sonderzeichen, die nicht gezählt werden, an alle möglichen Stellen einfügt, oder Buchstaben streicht.

Auf diesem Gebiet können interessante Erfahrungen gesammelt werden, indem ein solches Erkennungsprogramm selbständig entwickelt wird.

9.12 Einsatz von Programmiersprachen auf dem Mikrocomputer

In der Computertechnik werden verschiedene Typen von Programmiersprachen unterschieden. Da ist zunächst die Maschinensprache. Sie ist für den Computer direkt verständlich ausgeführt. Dann gibt es die maschinenorientierten Sprachen. Sie orientieren sich an der Maschinensprache, doch verwenden sie anstatt des Maschinencodes Ausdrücke, wie sie für den Menschen leichter merkbar sind.

Und schließlich gibt es noch die problemorientierten Sprachen. Mit ihnen ist es mög-

lich, Aufgabenstellungen zu formulieren, deren Text unabhängig von einem speziellen Computer ist. Zunächst soll die maschinenorientierte Sprache betrachtet werden.

9.12.1 Assembler

Die in den vorhergehenden Abschnitten behandelten Programme wurden bisher immer zunächst in der maschinenorientierten Version besprochen. Befehle wie MOV A, B oder ADD C sind solche maschinenorientierte Befehle. Eine Übersetzung dieser Befehle wurde bisher immer von Hand durchgeführt. Nun ist aber auch der Computer selbst dazu in der Lage. Dafür gibt es sogenannte Assembler. Sie haben die Aufgabe, die maschinenorientierte Sprache in die Maschinensprache umzusetzen. Auch für Mikrocomputer sind solche Assembler erhältlich. Man unterscheidet dabei zwei verschiedene Gruppen, die „Cross Assembler" und die „Resident Assembler". Der Unterschied besteht dabei im folgenden: Bei den „Cross Assemblern" wird der Assembler in der Programmiersprache, z. B. Fortran geschrieben, und das Programm wird dann auf einem großen Rechner gestartet.

Die „Resident Assembler" dagegen können auf dem Mikrocomputer selbst gestartet werden und erlauben daher die Unabhängigkeit von einem großen Computer.

Nur für wenige Mikrocomputer werden „Resident Assembler" geliefert, die meisten Hersteller von Mikrocomputern liefern bis jetzt nur Cross-Assembler. Ein „Resident Assembler" benötigt außerdem einen großen Arbeitsspeicher von einigen KBytes.

9.12.2 Interpreter

In folgendem werden die problemorientierten Programmsprachen behandelt. Es gibt die unterschiedlichsten problemorientierten Programmiersprachen. Hier seien daher nur einige namentlich genannt, wie Basic, Fortran, Algol 60, Algol 68, Cobol, PL/1. Für den Mikrocomputer 8080 existiert ein Programm in ROMs, das den Einsatz der Programmiersprache Basic auf dem Mikrocomputer erlaubt. Basic ist eine leicht erlernbare Programmiersprache, die für die Formulierung technisch-wissenschaftlicher Probleme geeignet ist. Ein Interpreter hat die Aufgabe, eine solche problemorientierte Programmiersprache in die Maschinensprache umzusetzen. Dabei wird das in der problemorientierten Programmiersprache geschriebene Programm aber nicht auf einmal übersetzt und dann in der Maschinensprache gestartet, sondern es wird eine „Aktion" übersetzt und dann ausgeführt, dann die nächste „Aktion" übersetzt und ausgeführt usw. Ein Interpreter ist einfacher aufgebaut als ein Compiler, bringt aber den Nachteil, daß die Ausführungszeit des Programms sehr groß wird. Dem wird ein Programmteil z. B. für ein Näherungsverfahren oft wiederholt, so wird das Programmteil trotz der Wiederholung neu übersetzt.

Um das zu vermeiden, werden Compiler angewendet.

9.12.3 Compiler

Der Compiler übersetzt ein Programm in einer problemorientierten Programmiersprache auf einmal in den Maschinencode, der dann auf dem Computer gestartet werden kann.

Für Mikrocomputer werden wieder zwei verschiedene Formen von Compiler unterschieden: „Cross Compiler" und „Resident Compiler". Die Firma Intel liefert für den 8080 und den 8008 einen solchen Cross-Compiler, der in der Programmiersprache Fortran abgefaßt ist, und der die problemorientierte Programmiersprache PL/M compilieren kann. Dabei stellt die Programmiersprache PL/M eine Teilmenge der Programmiersprache PL/1 dar. Außerdem enthält die Programmierspra-

che PL/M noch einige Zusatzbefehle, die in der Programmiersprache PL/1 nicht enthalten sind. Sie sind aber nötig, um den Befehlssatz des Mikrocomputers besser auszunützen. So enthält sie beispielsweise zusätzliche Befehle für die Interruptverarbeitung und für die I/O-Befehle (IN, OUT).

Abb. 9.12.3-1 zeigt ein in PL/M geschriebenes Programm, wie es dem Handbuch für PL/M entnommen wurde. Es sei noch gesagt, daß dieses Programm mit dem Namen EQUAL die Aufgabe hat, zwei Zeichenfolgen zu vergleichen und in Abhängigkeit des Ergebnisses den Wert 0 oder den Wert 0FFH auszugeben.

0FFH ist hier statt nur FFH erforderlich, weil bei der Sprache PL/M laut Definition Konstanten mit einer Ziffer beginnen müssen.

```
EQUAL: PROCEDURE (PTR1, PTR2) BYTE;

    DECLARE (PTR1, PTR2) ADDRESS;
    DECLARE (STRING1 BASED PTR1,
             STRING2 BASED PTR2) BYTE;
    DECLARE I ADDRESS, (J1, J2) BYTE;

    J1, J2, I = 0;
    DO WHILE J1=J2;
       IF J1=0FFH THEN RETURN 0FFH;
       J1 = STRING1(I);
       J2 = STRING2(I);
       I = I+1;
    END;
    RETURN 0;

END EQUAL;
```

Abb. 9.12.3-1 Ein Programm in PL/M geschrieben

F ist zwar im hexadezimalen Zahlensystem eine Ziffer, nicht aber in der genannten Sprache als solche definiert.

10 Anhang

10.1 Terminologie

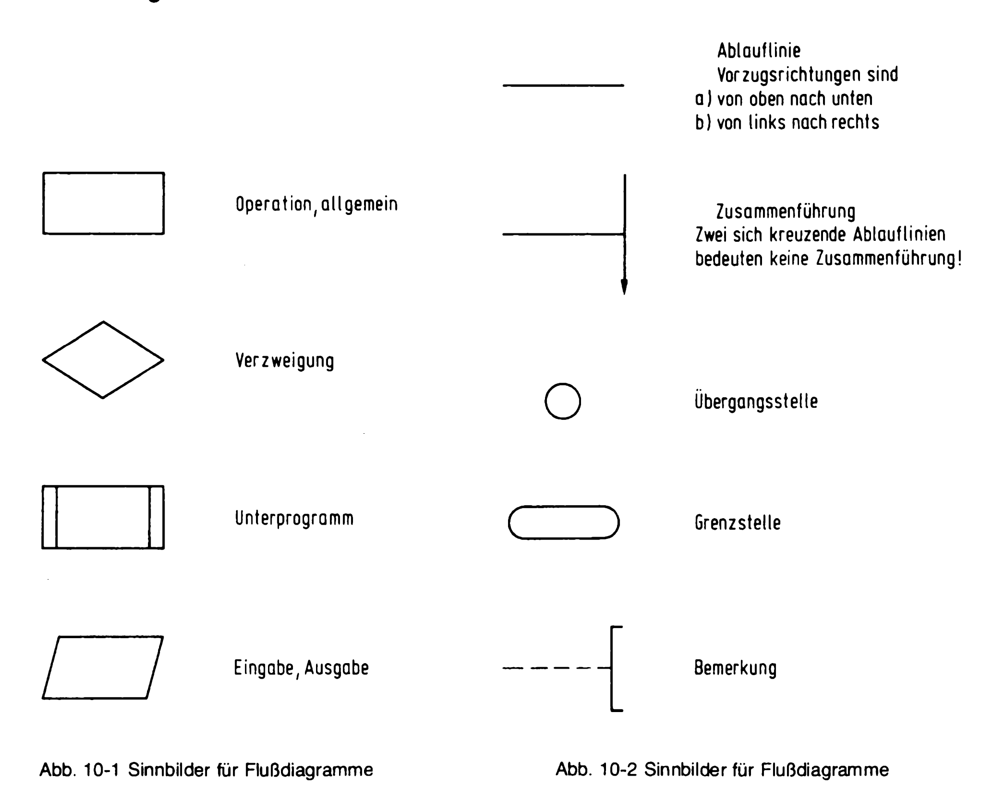


Abb. 10-1 Sinnbilder für Flußdiagramme

Abb. 10-2 Sinnbilder für Flußdiagramme

Tabelle 10-1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015
10	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031
20	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047
30	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063
40	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079
50	080	081	082	083	084	085	086	087	088	089	090	091	092	093	094	095
60	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111
70	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
80	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
90	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D0	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

10.2 Literaturverzeichnis

[1] Klein, Rolf-Dieter: Prüfstift für logische Schaltungen, FUNKSCHAU, Heft 2 (1975), S. 80.

[2] Klein, Rolf-Dieter: Gleichzeitige Anzeige aller IC-Pegel mit einem Oszilloskop. ELEKTRONIK, Heft 10 (1976), S. 88...89.

[3] TI Applikationsbuch, Band 1. Texas Instruments Deutschland GmbH.

[4] Franz Morat KG: TV-Computersystem 6800, Franz Morat KG, Eisenbach/Hochschwarzwald, 1976.

Elektronik: Alles über Mikroprozessoren. Sonderausgabe, Franzis-Verlag, 1976.

Intel: Datenkatalog 1976. Intel, 1976.

Löbel, Müller, Schmid: EDV-Taschenlexikon. 3. Auflage, Verlag Moderne Industrie, 1970.

Pelka: Was ist ein Mikroprozessor? Franzis-Verlag GmbH, 1976.

Siemens: Datenbuch 1976/77, Mikroprozessor-Bausteine System SAB 8080, Siemens, München.

Siemens: Mikroprozessoren und Mikrocomputer. Siemens, München.

10.3 Fachausdrücke – Glossar

A

Access: Zugriff. Möglichkeit zum Beispiel zu einer bestimmten Speicherzelle zuzugreifen.
A/D Umsetzer: Analog-Digital Umsetzer.
Adresse: Eine Bezeichnung für einen bestimmten Speicherplatz oder eines Speicherbereiches.
Akkumulator: Es handelt sich um ein Register, über das arithmetische und meist auch logische Befehle ausgeführt werden können. Der Akkumulator hat direkt Verbindung mit dem Rechenwerk.
Algol: Algorithmic Language. Es handelt sich um eine Programmiersprache für technisch-wissenschaftliche Probleme.
Alu: Arithmetic Logic Unit. Rechenwerk. In diesem Teil eines Rechners werden die arithmetischen und logischen Operationen ausgeführt.
ASCII: American Standard Code für Information Interchange. Ein häufig gebrachter Code für Informationsübertragung. Auch mit ISO-7-Bit-Code bezeichnet. (DIN 66 003).
Assembler: Ein Übersetzungsprogramm, das eine maschinenorientierte Programmsprache in die Maschinensprache übersetzt.
Assoziativ-Speicher: Ein Speicher, bei dem der Zugriff nicht über eine bestimmte Adresse, sondern über den Speicherinhalt erfolgt. Derartige Speicher sind bisher nur mit kleiner Kapazität realisiert.
Asynchron: taktunabhängige Arbeitsweise.

B

BASIC: Beginners All purpose Symbolic Instruction Code. Eine dialogorientierte, einfach zu erlernende höhere Programmiersprache, die auch für Mikrocomputer erhältlich ist.
Baud: Unter der Baudrate versteht man diejenige Frequenz mit der bei einem asynchronen seriellen Signal die einzelnen Bits übertragen werden (also auch Stopbits und Startbits). Die eigentliche Datenübertragungsfrequenz liegt meist etwas niedriger als die entsprechende Baudrate.
Bedingter Befehl: Ein Befehl, der in Abhängigkeit von bestimmten Zuständen von Registern o.ä. ausgeführt, oder nicht ausgeführt wird.
Betriebssystem: Darunter versteht man eine Reihe von Programmen, die es dem Computer ermöglichen selbständig Programme zu bearbeiten.
Bit: Binary Digit. Kleinste Informationseinheit.
Bootstrap-loader: Urlader. Ein Programm, das es dem Computer ermöglicht, Programm zu laden.
Borrow: negativer Übertrag.
Branch: Verzweigung. Siehe auch Sprung.
Breakpoint: Unterbrechungspunkt.
Buffer: Puffer. Speicher in dem Daten kurzfristig zwischengespeichert werden.
Bus: Sammelleitung, an der mehrere Bausteine angeschlossen sind. (Z.B. Adreßbus, Datenbus, Kontrollbus)
Byte: Damit sind 8 Bits gemeint.

C

CALL: Aufruf z.B. eines Unterprogramms. Siehe auch Unterprogramm.
Carry: Übertrag. Nicht mit Überlauf zu verwechseln.
Cartridge: Magnetbandkassette.
Clock: Takt.
CMOS: Complementary MOS. MOS-Technologie mit P- und N-Kanal Transistoren, die sich durch besonders geringen Ruhestrom auszeichnet.
COBOL: Common Busines Oriented Language. Eine Programmiersprache für vorwiegend kaufmännische Aufgaben.
Compiler: Ein Übersetzungsprogramm, das eine höhere Programmiersprache in den Maschinencode übersetzt.
Complement: Ergänzung.
Computer: Programmgesteuerte Rechenanlage.
Conditional: Bedingt.
Conversion: Übersetzung, Umkodierung.
Core: (Magnet-)Kern (Speicher).
CPE: Central Processing Element. Siehe Slice.
CPU: Central Processing Unit. Zentraleinheit. Rechenwerk und Steuerwerk eines Computers. Häufig wird der Mikroprozessor so bezeichnet.
Cross-Assembler: Ein Assembler für einen Mikrocomputer, der selbst auf einer Großrechenanlage läuft (oder auf einem Minicomputer).
Cross-Compiler: Ein Compiler für Mikrocomputer, der selbst auf einer Großrechenanlage läuft (oder auf einem Minicomputer).
CRT-Terminal: Cathode Ray Tube. Datensichtgerät.

D

Data-Bus: Datensammelschiene, siehe Bus.
Debugging: „Ent-Wanzen" Fehlersuche und Beseitigung.
Decrement: Schrittweises Erniedrigen um einen bestimmten Wert. Meist ist das Erniedrigen um den Wert 1 gemeint.
Density: Dichte.
Device: Gerät, Einheit.
D/A-Umsetzer: Digital-Analog Umsetzer.
Dialoggerät: Gerät zur direkten Datenein- und Ausgabe.
Digit: Ziffer, Stelle.
Direct-access: Direkter Zugriff.
Display: Anzeige.
DMA: Direct Memory Access. Speicherzugriff, bei dem die Information nicht über die CPU geht. Wichtig für die rasche Datenein- und Ausgabe.
DOS: Disk Operating System. Ein Programm, das es ermöglicht, mit einer Floppy-Disk zu arbeiten. Siehe auch Floppy-Disk.
Drum storage: Trommelspeicher.
Dump: Auszug eines Speicherinhalts.
Durchsatz: Anzahl der Operationen, die ein Computer in einer Zeiteinheit leistet.
Dynamischer Speicher: Bei einem solchen Speichertyp muß die Information zyklisch aufgefrischt werden. Vorteil ist die Verfügbarkeit von hohen Speicherkapazitäten.

E

EAROM: Electrical Alterable Read Only Memory. Festwertspeicher, dessen Inhalt sich elektrisch verändern läßt.
EBCDI-Code: Extended Binary Coded Decimal Interchange-Code. Ein alphanumerischer 8 Bit Code.
Editor: Ein Programm zur Änderung, Korrektur und Ausgabe von Anwenderprogrammen.
Emulation: Softwaremäßige Nachbildung eines Computers.
Enable: Freigabe.
EPROM: Erasable Programmable Read Only Memory. Ein mit ultraviolettem Licht löschbarer programmierbarer Festwertspeicher.
to erase: löschen.
Europakarte: Leiterplatte mit dem genormten Format: 100 mm x 160 mm.
Even-odd parity: gerade Parität oder ungerade Parität.
Exorciser: Hilfsgerät zur Entwicklung von Mikrocomputersystemen.

F

Fan-in: Eingangslastfaktor
Fan-out: Ausgangslastfaktor. Er gibt an, wieviele Bausteine einer gleichen Logikserie an einen Ausgang mit dem angegebenen Fan-out angeschlossen werden können.
Fifo: First In First Out. Zuerst eingehende Daten werden auch zuerst wieder ausgegeben.
Fixed-point: Festkomma.
Flag: Ein Flip-Flop, das als Zustandsanzeiger verwendet wird (z.B. für Überlauf, Parität etc.)
Flip-Flop: Bistabiles Speicherelement, das 1 Bit speichern kann.
Floating-point: Gleitkomma.
Floppy Disk: Ein billiger Plattenspeicher, der auch für Mikrocomputer entwickelt wurde.
Fortran: Formula Translation. Eine problemorientierte Programmiersprache, die für technisch-wissenschaftliche Probleme ausgelegt ist. Für Mikrocomputer (zur Zeit für 8080) ist schon ein residenter Compiler lieferbar.
Frontpanel: Bedienungsfeld.

G

Gate: Verknüpfungsschaltung.

H

Handler: Routine zur Kontrolle eines peripheren Gerätes.
Handshaking: Quittungsbetrieb. Methode um Geräte mit verschiedenen Arbeitsgeschwindigkeiten durch Austausch von Steuersignalen zu synchronisieren.
Hardware: Damit sind alle Geräte, Bauteile eines Systems gemeint.
hexadezimal: Siehe Sedezimal.
HIGH: Logikpegel nach DIN 41785. Bei positiver Logik entspricht „H" der „logischen" Eins.
High order: Höherwertige Stelle.
HLL: High Level Logic. Entsprechend wie die Siemens LSL-Serie (Langsame Störsichere Logik).

I

Increment: Schrittweises Erhöhen um einen bestimmten Wert (meist um eins).
Inhibit: Sperr . . .
Input: Eingabe.
Instruction: Befehl. Anweisung.
Interface: Schnittstelle. Mit Hilfe eines Interface können zwei Systeme einander angepaßt werden. „Interface" kann auch mit „Anpaßschaltung" übersetzt werden.
Interpreter: Ein Interpreter ist ein Programm, das die Befehle einer höheren Programmiersprache (z.B. Basic) direkt ausführt und nicht zuerst das Programm in einen Maschinencode umwandelt.
Interrupt: Unterbrechung. Durch einen Interrupt, den meist ein externes Peripheriegerät anfordert, wird die laufende Programmausführung unterbrochen und eine spezielle Unterbrechungsroutine ausgeführt. Danach erfolgt ein Rücksprung in das laufende Programm.

Ion Implantation: Ein Dotierverfahren, das niedrige Schwell- und Versorgungsspannungen ermöglicht, sowie hohe Packungsdichten erlaubt.
TRI-state: Ein Ausgang, der drei verschiedene Zustände annehmen kann. Entweder ist der Ausgang auf HIGH oder auf LOW oder offen. Derartige Ausgänge sind besonders für Bus-Systeme geeignet.

J

Jump: Sprung. Siehe Sprung.

K

Keyboard: Tastatur.
Kit: Bausatz.
Kompatibel: austauschbar, aneinander angepaßt.

L

Label: Marke. In Programmiersprachen ist damit meist eine symbolische Adresse gemeint. Sonst, z.B. bei Magnetbändern, ist damit ein Identitätskennzeichen gemeint.
LED: Light Emitting Diode. Licht ausstrahlende Halbleiterdiode.
Lifo: Last In First Out. Zuletzt gespeicherte Daten werden zuerst ausgegeben.
Listing: Ausdruck. Auflistung.
Loader: Ein Ladeprogramm.
Logic Analyzer: Ein Hilfsgerät zum Testen von umfangreichen Digitalschaltungen, mit einer Anzeigemöglichkeit für die logischen Zustände in dieser Schaltung.
Loop: Schleife. Durch einen Sprung zurück kann zum Beispiel eine Schleife entstehen.
Low order: niederwertige Stelle.
Low-power-TTL: Schaltkreise mit gegenüber der normalen TTL Serie stark verringertem Leistungsverbrauch.
LSI: Large Scale Integration. Eine hohe Integrationsdichte.
LSL: Langsame Störsichere Logik.

M

Makro: Darunter wird eine Folge von Befehlen verstanden, die der Programmierer zunächst definiert. Später kann der Programmierer diese Befehlsfolge mit Verwendung eines symbolischen Bezuges automatisch an verschiedene Stellen seines Programmes durch den Assembler einfügen lassen.
Maschinencode: Maschinensprache. Damit ist ein Binär-Code gemeint, der vom Mikrocomputer direkt verstanden wird.
Maske: Ein Bitmuster, mit dem bestimmte Bitgruppen ausgeblendet oder komplementiert werden können.
Memory: Speicher.
Microprogrammierbar: Der Befehlssatz eines Prozessors kann mit Hilfe von Mikrobefehlen definiert werden.
Mikrocomputer: Besteht aus einem Mikroprozessor, Speichern und Peripherie.
Mikroprozessor: Ein integrierter Baustein, als Teil eines Mikrocomputers, der ein Leit- und ein Rechenwerk besitzt. Der interne Ablauf kann in der Regel von außen durch Software beeinflußt werden.
MNOS: Metall Nitrod Oxid Semiconductor. Technologie zum Aufbau von EAROMs.
Modem: Modulator und Demodulator. Eine Schaltung, die Daten für die Fernübertragung aufbereitet.
Monitor: Ein Programmsteuersystem, das auch aus Hardware bestehen kann.
MOS: Metall Oxyd Semiconductor. Eine Halbleitertechnologie, die es ermöglicht sehr hohe Schaltungs-Eingangswiderstände zu realisieren.
MSI: Medium Scale Integration. Mittlerer Integrationsgrad (weniger als 100 Verknüpfungen).
Multiplex: Übertragung von mehreren verschiedenen Informationen, die dazu zeitlich hintereinander übertragen werden.
Multiprocessing: Ein Computer, der Probleme mit Hilfe von mehreren CPUs löst.

N

Nesting: Verschachtelung. Zum Beispiel verschachteln von Interrupts.
N-Kanal-MOS: MOS-Technologie, die mittelschnelle Schaltgeschwindigkeiten erlaubt.

O

Off-line: Der Benutzer ist dabei nicht hardwaremäßig mit dem Computer verbunden, sondern der Verkehr wird über Datenträger abgewickelt.
Oktal: Zahlendarstellung mit 8 verschiedenen Grundelementen (0, 1, 2, 3, 4, 5, 6, 7).
On-line: Dabei ist das Terminal des Benutzers über eine Datenleitung direkt mit dem Computer verbunden.
Output: Ausgabe.
Overflow: Überlauf.

P

Packen: Dabei werden zum Beispiel zwei Dezimalzahlen in einem Byte untergebracht.
Parity: Parität. Gleichheit.
Pass: Lauf. Zum Beispiel eines Programms.
Peripherie: Externe Datenend- und Speichergeräte.
PIA: Peripheral Interface Adapter. Ein Baustein, der den Ein- und Ausgabeverkehr zwischen dem Mikroprozessor und der Peripherie abwickelt.
Pin-kompatibel: Alle Anschlüsse des Bausteins sind gleich, so daß ein direkter Austausch möglich ist, doch kann die interne Schaltung unterschiedlich sein.

Pipelining: Fließbandverarbeitung. Durch diese Verarbeitungsform kann die Ausführungszeit stark verkürzt werden. Während ein Befehl gerade ausgeführt wird, wird der nächste Befehl schon geholt. Bei Sprungbefehlen ergeben sich allerdings zusätzliche Verzögerungen. (Der Mikroprozessor 6502 (KIM) ist z.B. eine solche Pipelining-Maschine)
P-Kanal-MOS: Relativ langsame MOS-Technologie.
PL/1: Programming Language 1. Eine höhere Programmiersprache.
PLA: Programmable Logic Array. Eine programmierbare Logikanordnung, die aus ROMs besteht. Es lassen sich damit zum Beispiel Codeumwandlungen verwirklichen. Die Matrix gliedert sich in drei Komplexe. Produkttermmatrix (UND-Verknüpfungen). ODER-Matrix. Programmierteil der Ausgänge.
PL/M: Programming Language for Mikrocomputers. Eine höhere Programmiersprache für Mikrocomputer, die auf der Sprache PL/1 basiert.
Pointer: Zeiger. Ein Speicherplatz, der eine Adresse enthält. Mit einem Zeiger lassen sich leicht Stacks aufbauen (Stack-Pointer).
Programm: Ein Programm ist eine Folge von Anweisungen (Befehlen), die zur Lösung eines bestimmten Problems dienen sollen.
Programmiersprachen: Eine Sprache zur Formulierung von Programmen, die automatisch in die Maschinensprache übersetzt werden können.
PROM: Programmable Read Only Memory. Ein programmierbarer Festwertspeicher.
to punch: stanzen, lochen.

R

RALU: Register and Arithmetic Logic Unit. Ein Prozessorelement mit einer ALU und einigen Registern.
RAM: Random Access Memory. Ein Schreib-/-Lesespeicher mit wahlfreiem Zugriff.
Reader: (Lochstreifen- oder Lochkarten-)Leser.
Real-Time: Echtzeit. Arbeitsweise eines Computers.
Redundanz: Teil einer Nachricht, der zur eigentlichen Information nichts mehr beiträgt.
Refresh: Wiederauffrischung. Wird bei dynamischen Speichern benötigt, um einen Informationsverlust zu verhindern.
Relocalisierbar: Ein Programm, das auf verschiedenen Speicherbereichen direkt lauffähig ist, heißt relokalisierbar.
REPROM: Reprogrammable Read Only Memory. Ein Festwertspeicher, der sich löschen und wieder neu programmieren läßt.
Request: Anfordern. Anforderung.
to reset: rücksetzen.
Resident-Assembler: Ein Assembler, der auf der Maschine selbst läuft. Siehe Cross-Assembler.
Resident-Compiler: Ein Compiler, der auf der Maschine selbst läuft. Siehe Cross-Compiler.
to rewind: Zurückspulen. Zum Beispiel von Bandgeräten.
ROM: Read Only Memory. Ein Festwertspeicher von dem nur gelesen werden kann, mit wahlfreiem Zugriff.
Run: Durchlauf.

S

to scan: abtasten.
Schnittstelle: Pegel- und Anschlußgenormte Trennstelle zwischen zwei Geräten.
sedezimal: Zahlendarstellung mit 16 verschiedenen Grundelementen. (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). Häufig auch mit Hexadezimalsystem bezeichnet.
to select: auswählen.
to sense: abtasten.
Sign: Vorzeichen.
Simulator: Ein Programm, das einen Mikrocomputer auf einer anderen Rechenanlage zu simulieren gestattet.
Slice: Prozessorelement. Damit ist es möglich, durch Zusammenschalten von mehreren solchen Elementen einen Mikroprozessor von beliebiger Wortlänge aufzubauen.
Software: Hierunter versteht man unter anderem alle Arten von Programmen.
SOS: Silicon On Sapphire. Neue schnelle MOS-Technologie.
Source: Quelle.
Space: Freiraum.
Sprung: Mit einem Sprung kann die lineare Abarbeitung eines Programms verlassen werden.
SSI: Small Scale Integration.
Stack: Stapelspeicher, Kellerregister. Merkmal für einen Stack ist, daß eine Informationseinheit immer nur an der Stelle entnommen werden kann, an der sie gerade hinzugefügt werden konnte. Siehe auch LIFO.
State: Zustand. Operationsschritt.
Statement: Anweisung. Befehl.
statischer Speicher: Ein Speicherbaustein, der keinen Wiederauffrischzyklus benötigt.
Steuerwerk: Dieser Teil eines Computers kontrolliert die Ausführung sämtlicher Befehle. Wird auch mit Leitwerk bezeichnet.
Subroutine: siehe Unterprogramm.
Supervisor: Ein Organisationsprogramm.
Synchron: Ein Takt steuert den genauen Ablauf.

T

Tape: Ein Magnetband oder ein Lochstreifen.
Terminal: Datenendstation. Ein Gerät zur Datenein- und/oder Datenausgabe.

Text-Editor: siehe Editor.
Time sharing: Zeitscheibenverfahren. Ein Verfahren, bei dem mehrere Benutzer im On-Line Betrieb auf einen Computer zugreifen können.
Trace: Ablaufverfolgung. Ein Programm kann durch die schrittweise Ausführung und Protokollierung überwacht und so ein eventueller Fehler leichter gefunden werden.
Track: Spur. Bahn.
to transfer: übertragen.
TTY: Teletype. Fernschreiber.
two pass assembler: Ein Assembler, der die Übersetzung in zwei Durchläufen vornimmt. Die Quelle wird also zweimal eingelesen.

U

UART: Universal Asynchronous Receiver/Transmitter. Man versteht darunter eine Schaltung, die sowohl die Parallel-Serien Umsetzung als auch die Serien-Parallel Umsetzung für asynchronen Datenverkehr vornimmt.
Mnemonischer Code: Leicht zu merkende Kurzwörter, deren Inhalt auf die Verwendung schliessen läßt. Derartige Kurzwörter werden in Assemblersprachen angewendet.
Unit: Gerät. Einheit.
Unterprogramm: Gleiche Befehlsfolgen, die in einem Programm an verschiedenen Stellen benötigt werden, können zum Beispiel als Unterprogramm ausgeführt werden. Es ist dann möglich dieses Unterprogramm vom Hauptprogramm aus aufzurufen. Vorteil der Unterprogrammtechnik ist Speicherersparnis.

V

Valid: gültig.
Vektor Interrupt: Ein Interrupt, bei dem jedes anfordernde Gerät mit einer eigenen Unterbrechungsroutine bedient wird.
VLSI: Very Large Scale Integration. Sehr hoher Integrationsfaktor.
Volatile: Flüchtig.

W

Worst case: ungünstigster Fall.
Wort: Zusammenfassung mehrerer Bits, sie können meist zusammenhängend verarbeitet werden (8-64 Bits).

Z

Zugriff: Zugang zum Beispiel zu einer bestimmten Speicherzelle.
Zyklus: Eine Anzahl von Schritten, die wiederholt werden und im Ablauf gewisse Ähnlichkeiten aufweisen.

Sachverzeichnis

F

G

H

I

J

K

L

M

N

O

W

X

Z

Weitere Franzis Elektronik-Fachbücher

Analoge integrierte Schaltungen

Ein Lehrbuch, Schaltungen mit Operationsverstärkern und analogen Multiplizierern zu entwerfen. Von Miklós Herpy. Das Werk behandelt ausführlich den inneren Schaltungsaufbau sowie die Anwendung von monolithischen integrierten Operationsverstärkern und analogen Multiplizierern.

Das Werk lehrt ferner, neue Schaltungen nach neuartigen Methoden zu entwerfen. Dazu wird dem Entwickler und dem Konstrukteur nahegelegt, jetzt in Schaltungs- und Funktionseinheiten zu denken, wogegen die elektronische Stufe und die Teilschaltung nur noch eine untergeordnete Rolle spielen. Das setzt zweifellos etwas mathematisches Arbeiten voraus. Da zeigt sich der Autor – ein Hochschullehrer – als guter Pädagoge. Das Entwickeln und das Ableiten der allgemeingültigen Gesetze wird ausreichend erklärt und geübt und damit bald zur reinen Routine. Das Werk lenkt auf die Schwerpunkte bei der Beurteilung, bei dem Entwurf und bei der Anwendung von analogen integrierten Schaltungen.

522 Seiten mit 373 Abbildungen und 51 Tabellen. Leinen geb. DM 68.–

ISBN 3-7723-6151-X

Von der Schaltalgebra zum Mikroprozessor

Die Mikroprozessoren und ihre festverdrahtete und programmierbare Logik. Von Horst Pelka. Mathematische Logik und elektronische Technik ergeben einen Mikroprozessor. Hier sind die Grundlagen dazu umfassend und doch kompakt dargestellt. Ausgegangen wird von den binären Zahlensystemen und Codes, um so in die Grundlagen der Digitaltechnik einzudringen. Auf die verschiedenen bipolaren und MOS-Technologien integrierter Schaltungen wird ebenso eingegangen, wie auf die schaltungstechnische Realisierung von Verknüpfungsgliedern, Flip-Flops, Schieberegister und Zeitschaltungen. Fast die Hälfte des Buches behandelt die Grundlagen und Programmierung von Mikroprozessoren. Der Stoff ist einfach und klar dargestellt, viele Programmbeispiele erleichtern das Verständnis. Mit diesem Buch lernt der Leser das Gebiet der festverdrahteten Logik und das der Mikroprozessoren kennen. Es ist ihm möglich, Entscheidungen bei der Auswahl dieser festverdrahteten und programmierbaren Logik zu treffen.

304 Seiten mit 170 Abbildungen und 8 Tabellen.
Lwstr-kart. DM 26.80

ISBN 3-7723-6421-7

Schaltungen und Bausteine der Elektronik

Eine katalogartige Übersicht elektronischer Grundschaltungen, vom Transistor bis zum Mikroprozessor. Von Horst Pelka. Straff, präzise und übersichtlich ist hier ein Status der modernen elektronischen Bausteine und Schaltungen aufgestellt worden. Das war notwendig, weil der Trend zu einer immer höheren Integrierung geht. Es sei nur an die letzten Höhepunkte Operationsverstärker und Mikroprozessor erinnert. Der Lernende findet sich nun in dem weiten Feld der angewandten Elektronik gut zurecht, weil technischer Ballast abgeworfen wurde. Der Praktiker findet hier den Ausgangspunkt, von dem aus er die Elektronik in Techniken einführt, die bisher noch nicht von ihr berührt worden sind. „So könnte man es machen!" wird mancher Benutzer erleichtert sagen, wenn er dieses Praktikum um Rat gefragt hat.

160 Seiten, 155 Abbildungen.
Lwstr-kart. DM 16.80

ISBN 3-7723-6361-X

Franzis-Verlag, München

Basteln für Ingenieure

Kaum zu glauben, daß ein Mikrocomputer im Selbstbau hergestellt werden kann! Daß dieses Vorhaben glückte, hat der Autor bewiesen. Wie ein hinreichend ausgebildeter Elektroniker das nachvollziehen kann, wird in dem Buch hier dargestellt.

Zunächst muß die Hardware geschaffen werden. Eingabetastatur, Mikroprozessor, Speicher verschiedener Art, Drucker, Sichtgerät, das alles muß zu einer funktionierenden Einheit zusammengeschlossen werden. Und das geht. Es geht sogar mit preiswerten, modernen Teilen, die in den einschlägigen Fachhandlungen zu haben sind.

Nun die Software. Da zeigt der Autor mehrere Möglichkeiten auf. Nicht etwa nur ein kleines Programm, das immer wieder stupide abläuft. Nein, ausführliche Programme werden vorgestellt, die zahlreiche Spiele, mathematische Aufgaben, wissenschaftliche Probleme bearbeiten können.

Als Abschluß und Höhepunkt fügt der Autor Anregungen hinzu, selbst Programme zu schreiben und in dem eigenen Mikrocomputer zu erproben. Was will man mehr?



Rolf-Dieter Klein
Mikrocomputersysteme

Franzis

ISBN 3-7723-6381-4